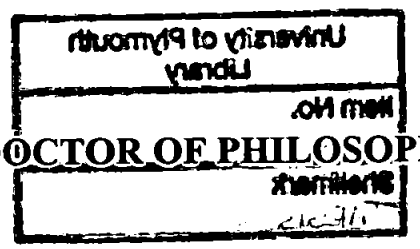
An Interactive Visualisation System for Engineering Design using Evolutionary Computing

by

IAN STEPHEN JAMES PACKHAM

A thesis submitted to the University of Plymouth
in partial fulfilment for the degree of



DOCTOR OF PHILOSOPHY

School of Computing, Communications and Electronics
Faculty of Technology
University of Plymouth

December 2003

An Interactive Visualisation System for Engineering Design using Evolutionary Computing

Ian Stephen James Packham

Abstract

This thesis describes a system designed to promote collaboration between the human and computer during engineering design tasks. Evolutionary algorithms (in particular the genetic algorithm) can find good solutions to engineering design problems in a small number of iterations, but a review of the interactive evolutionary computing literature reveals that users would benefit from understanding the design space and having the freedom to direct the search. The main objective of this research is to fulfil a dual requirement: the computer should generate data and analyse the design space to identify high performing regions in terms of the quality and robustness of solutions, while at the same time the user should be allowed to interact with the data and use their experience and the information provided to guide the search inside and outside regions already found.

To achieve these goals a flexible user interface was developed that links and clarifies the research fields of evolutionary computing, interactive engineering design and multivariate visualisation. A number of accessible visualisation techniques were incorporated into the system. An innovative algorithm based on univariate kernel density estimation is introduced that quickly identifies the relevant clusters in the data from the point of view of the original design variables or a natural coordinate system such as the principal or independent components. The robustness of solutions inside a region can be investigated by novel use of 'negative' genetic algorithm search to find the worst case scenario. New high performance regions can be discovered in further runs of the evolutionary algorithm; penalty functions are used to avoid previously found regions. The clustering procedure was also successfully applied to multiobjective problems and used to force the genetic algorithm to find desired solutions in the trade-off between objectives.

The system was evaluated by a small number of users who were asked to solve simulated engineering design scenarios by finding and comparing robust regions in artificial test functions. Empirical comparison with benchmark algorithms was inconclusive but it was shown that even a devoted hybrid algorithm needs help to solve a design task. A critical analysis of the feedback and results suggested modifications to the clustering algorithm and a more practical way to evaluate the robustness of solutions. The system was also shown to experienced engineers working on their real world problems, new solutions were found in pertinent regions of objective space; links to the artefact aided comparison of results. It was confirmed that in practice a lot of design knowledge is encoded into design problems but experienced engineers use subjective knowledge of the problem to make decisions and evaluate the robustness of solutions. So the full potential of the system was seen in its ability to support decision making by supplying a diverse range of alternative design options, thereby enabling knowledge discovery in a wide-ranging number of applications.

Contents

List of Figures

List of Tables

Software and Figure Acknowledgements

The system described in this thesis was written in MATLAB ® 6.1 by The MathWorks, Inc., Massachusetts, USA. Many of the figures were produced by the author using MATLAB:
www.mathworks.com

The system uses the GA Toolbox for Matlab (Chipperfield *et al.* 2002), published under the GNU General Public Licence (GNU 2003):
http://www.shef.ac.uk/~gaipp/ga-toolbox/

The MATLAB Toolbox for Density Estimation (Beardah & Baxter 1996, Beardah 1999) was used by the system and to generate most of the figures in Chapter 4, with the kind permission of Mike Beardah of The Nottingham Trent University, Nottingham, UK:
http://science.ntu.ac.uk/msor/ccb/densest.html

MATLAB code provided by the FastICA algorithm (Hyvärinen 2003) was used to undertake independent component analysis in the system and produced the results shown in Figures 3.15, 3.16, 3.18, 4.4, 4.11 and 5.7. The FastICA package is Copyright © 1998 by Jarmo Hurri, Hugo Gävert, Jaakko Särelä, and Aapo Hyvärinen, published under the GNU General Public License (GNU 2003):
http://www.cis.hut.fi/projects/ica/fastica/

Figure 5.8 was produced by the system using SAMMON mapping code. This software was contributed to the SOM Toolbox by Juha Vesanto. The SOM Toolbox (Vesanto *et al.* 2000) is Copyright © 2000 by Esa Alhoniemi, Johan Himberg, Juha Parhankangas and Juha Vesanto, published under the GNU General Public licence (GNU 2003):
http://www.cis.hut.fi/projects/somtoolbox/

Figures 3.6, 3.7, 3.11 were generated by XmdvTool (Ward 1994), a public-domain visualisation package developed at the Worcester Polytechnic Institute, Worcester, MA, USA, with the kind permission of Mathew Ward:
http://davis.wpi.edu/~xmdv

The interface screenshots shown in Figure 3.18 were from the GGobi system (Swayne *et al.* 2001), used with the kind permission of Deborah Swayne at AT&T Labs – Research, New Jersey, USA and Dianne Cook of the Iowa State University, Iowa, USA:
www.ggobi.org

The dendograms in Figures 3.19 were produced using MATLAB code kindly made available by Roger Jang at Tsing Hua University, Taiwan (Jang 2003):
http://neural.cs.nthu.edu.tw/jang/matlab/demo/

Figures 3.23, 3.24 and 3.25 are found on the website "The Acquisition of Insight" (Spence 2003), used with the kind permission of Robert Spence of Imperial College, London, UK:
http://www.ee.ic.ac.uk/research/information/www/Bobs.html

Acknowledgements

This work was funded by the Engineering and Physical Sciences Research Council (EPSRC) through a Research Studentship converted to a CASE award by BAE SYSTEMS. Many thanks go to these institutions for their financial assistance.

The first supervisor for this research was Dr. Sue Denham, of the Centre for Theoretical and Computational Neuroscience, I am very grateful for all the support, direction, encouragement, enlightening comments and infinite patience that she provided. There have been a number of other supervisors involved in the project that I would also like to thank. Dr. Yaqub Rafiq, of the School of Engineering, for his immense technical knowledge, practical advice and enthusiasm. Dr. Ken Fisher, now of the London Metropolitan University, for the lengthy discussions and valuable feedback. Prof. Ian Parmee, now of the University of Western England, for the ideas that came out of discussions during the initial stages of the project.

Thanks also go to Mr. Martin Borthwick, Mr. David Easterbrook and Mr. Colin Southcombe, staff at the School of Engineering, who generously provided expert evaluation of the system giving much encouragement and advice. Further thanks to all the people who supplied constructive criticism and suggestions about the system, particularly the doctoral students who kindly agreed to undertake the user evaluation experiments, giving up much of their valuable time that was greatly appreciated: Mr. Stalin Munoz, Mr. Richard Holden, Miss. Pauline Campbell and Mr. Huck Turner. Many of the academic and support staff at the School of Computing, Communications and Electronics and the rest of The University of Plymouth also helped to make the completion of this thesis possible, in particular Prof. Pat Pearce, Mrs. Carole Watson and Mr. Peter Grebot.

I have made many friends since coming to Plymouth, both inside and outside the university. There are far too many to mention by name, but thanks to you all for helping me enjoy the good times and putting up with me in the difficult times. I also appreciate the support and encouragement from all my friends who live further away; I look forward to seeing more of you now I have finally completed this thesis. Of course this project would never have even started without the neverending faith, encouragement and backing of my parents and family – you were never far from my thoughts.

AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

This study was financed with the aid of a studentship from the Engineering and Physical Science Research Council (EPSRC) converted to CASE award by BAE SYSTEMS.

A programme of advanced study was undertaken, relevant seminars and conferences were regularly attended at which work was often presented. The author was an instructor on a master's level module in Computational Intelligence, entitled Adaptive Computing in Design and Manufacture.

Publications:

Packham I.S.J. and Parmee I.C. (2000)*, "Data Analysis and Visualisation of Cluster-Oriented GA Output", *Poster Proceedings of the Fourth International Conference on Adaptive Computing in Design and Manufacture (ACDM'00)*, Parmee I C (ed.), 26-28 April 2000, University of Plymouth, pp. 71-74.

Packham I.S.J. and Parmee I.C. (2000)*, "Data Analysis and Visualisation of Cluster-Oriented Genetic Algorithm Output", *IEEE International Conference on Information Visualization (IV2000)*, Banissi E., Bannatyne M., Chen C., Khosrowshahi F., Sarfraz M. and Ursyn A. (eds.), 19-21 July 2000, London, UK, IEEE Computer Society, pp 173-178.

Parmee I., Cvetković D., Bonham C. & Packham I. (2001), "Introducing Prototype Interactive Evolutionary Systems for Ill-Defined, Multi-Objective Design Environments", *Advances in Engineering Software*, Elsevier Science Ltd, Vol. 32, No. 6, pp. 429-441.

Packham I.S.J. and Denham S.L. (2003)*, "Visualisation Methods for Supporting the Exploration of High Dimensional Problem Spaces in Engineering Design", *Proceedings of the IEEE International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV2003)*, Roberts J.C. (ed.), London, UK, 15 July 2003, IEEE Computer Society, pp 2-13.

Packham I.S.J., Rafiq M.Y., Borthwick M.F. and Denham S.L., "Interactive Visualisation for Decision Support and Evaluation of Robustness using Evolutionary Computing", accepted for publication as a full paper at the *11th International Workshop for Intelligent Computing in Engineering*, Weimar, Germany, May/June 2004.

Packham I.S.J., Borthwick M.F., Rafiq M.Y. and Denham S.L., "Improved Understanding of Rainfall Runoff Modelling using Interactive Visualisation and Evolutionary Computing", accepted for publication as a poster at the *11th International Workshop for Intelligent Computing in Engineering*, Weimar, Germany, May/June 2004.

Rafiq M.Y., Packham I.S.J. Easterbrook D.J. & Denham S.L., "Visualising Search and Solution Spaces in the Optimum Design of Biaxial Columns", Submitted for publication to the *Journal of Computing in Civil Engineering*, ASCE.

* Paper presented by author

Other Conference Attended:

The Sixth International Conference on Artificial Intelligence in Design (AID'00), held in the Worcester Polytechnic Institute (WPI), Worcester, Massachusetts, USA, conference chair: John Gero, University of Sydney, Australia. Workshops and main conference attended on 24-29 June 2000.

Signed Ian Parkham

Date 11/11/2004

Chapter 1: Introduction

1.1 Motivation of Research

This research project was inspired by recognising the need to understand the information supplied by optimisation algorithms working on engineering design problems. Many algorithms, particularly evolutionary algorithms, have been successfully incorporated into systems that supply good solutions to practical problems. However many of those systems return 'the best' solution they can find without explaining where the solution comes from or providing alternative options. Furthermore if the designs produced by a system are not satisfactory, the engineer (or user) is forced to repeat the algorithm using different parameter settings to find another 'good' solution. Evaluating the robustness or sensitivity of the solution is usually limited to varying the input variables and observing the change in design objective or cost, leading to further blind search that may be inconclusive.

In practice an engineering design model does not fully explain the problem and engineers use their experience to choose between different solutions. As computational power continues to increase, more solutions can be generated to more complex and higher dimensional problems, increasing the diversity of choice. But accessing and understanding all the available choices is very difficult. Therefore a system that enables users to visualise and interact with high dimensional data was seen as an essential tool needed for modern engineering design activity (Parmee & Bonham 1998). Instead of returning a single solution to a problem, all the solutions that have been considered by the algorithm can be presented to the user for analysis. By interacting with the system the user can define a region of the search space and run a more detailed search or check the robustness of that region. Due to the large variety of solutions returned, an experienced engineer, or group of multidisciplinary engineers (Baird *et al.* 2000), can assess the merits of each solution and its neighbours and come to some decisions on the best set of design variables to use.

Before a system can evaluate solutions for engineering design purposes, or any other purpose, the data needs to be generated in an efficient way. For a long time evolutionary computing has been involved in engineering design because of its advanced search and fast optimisation capabilities; evolutionary computing is a group of techniques inspired by Darwin's theory of evolution (Darwin 1859). The genetic algorithm (GA) is one such approach proposed by Holland (1975) that explores the search space and exploits good solutions. The GA adapts to the objective function that defines the good solutions, so can be controlled in various ways by the user. Other search and hill climbing algorithms could be used, but the GA was chosen because of its more exploratory attributes (crossover and mutation), robustness, coding simplicity and the relatively small number of parameters required. The GA has been successfully applied to the theory of engineering design (Goldberg 1989, Fonseca & Fleming 1995, Deb *et al.* 2000) and used in many practical applications (see Parmee *et al.* 2000, Gen & Chen 1997, Rafiq & Southcombe 1998, Balling *et al.* 2000, Walters *et al.* 1999).

Most systems using genetic algorithms applied to an engineering problem merely return a few solutions to the problem without explaining where they came from or how they relate to each other; in a multidimensional problem this is particularly frustrating. Traditional approaches to evaluate the robustness of solutions in engineering design vary the input variables close to the desired solution and evaluate the change in objective value of those solutions to ensure that manufacturing tolerances are met. Taguchi analysis is a systematic approach to this process using orthogonal matrices (Taguchi 1986); more recent non-linear techniques can also be used to evaluate the robustness of the feasibility of solutions (Parkinson *et al.* 1993, Du & Chen 2000). Again there is an element of trial and error with these techniques because of the large size of the search space. In order to fully understand the problem there would be an obvious benefit from visualising all the data available and trying to understand how solutions and variables relate to each other.

Parmee *et al.* (2000) proposed a number of ways evolutionary algorithms could be used in an interactive engineering design environment, but the details of exactly how this would be performed were not given. Many researchers also advocate such systems (Mathews & Rafiq 1994) but usually implement them for specific problems (Mathews *et al.* 1996). Further review of the literature reveals that a lot of research has been undertaken in interactive evolutionary computation (see recent review in Takagi 2001), but the ability to directly interact with the search space, analyse the robustness of solutions and guide the system towards innovative solutions is limited.

1.2 Aims of Research and Implementation

This thesis explains the development, testing and evaluation of a visualisation system designed to support the iterative and systematic process of engineering design using evolutionary computing. The overall research aim was to promote collaboration between the human and computer to help improve understanding of the search space and how variables interact to form robust solutions. It was determined that the strengths of the human and computer would be exploited by supporting the following dual requirement:

1. The computer should quickly generate data and analyse the design space to identify high performing regions in terms of the quality and robustness of solutions
2. The user should be allowed to interact with the data and guide the search using their experience and the information provided

The multivariate visualisation, clustering and statistical analysis literature was consulted to discover techniques that fulfil this dual requirement. It was established that many state of the art systems use advanced techniques to classify data but the visual representation is often too abstract to describe a concrete engineering design problem; even principal component analysis is limited unless the results are related to the original design

variables. Conversely techniques that are specifically designed for interactive engineering design do represent the data in an accessible way (for example Tweedie *et al.* 1996b), but the data is generated randomly and there is no way of guaranteeing the robustness of solutions. Therefore a new system was designed to the following specifications:

a. Generate informative data as quickly as possible using the genetic algorithm (GA)
b. Quickly identify the main clusters in the data pertinent to engineering design and relate the clusters to the original design variables
c. Ability to analyse interesting regions of the search space suggested by the GA, in particular to evaluate the quality and robustness of those regions
d. Introduce multivariate visualisation techniques that are understandable and do not distort the data for engineering design purposes
e. The system should be easy to use and flexible to allow further GA search and analysis inside and outside regions already found

The system was then evaluated to test that it satisfied the dual requirement and was an improvement on automated algorithms. The preliminary evaluation was designed to test the usefulness and applicability of features of the system to help users understand engineering design problems. An engineering design task was simulated on continuous artificial test functions by asking novice users to find regions of the search space that are robust according to a tolerance specification. The test functions were designed so that the 'ideal' regions the user should be looking for could be determined. The ability of the system to support decision making was evaluated by asking the users to give a relative preference measure for the regions found. The quantitative success of the users was evaluated by comparing their results with those from benchmark algorithms used to solve problems with multiple optima. The overall success of the system was evaluated through a critical analysis of the results and feedback supplied by the participants.

Subsequently the system was shown to experienced engineers working on their design problems to assess its usefulness and effectiveness in the real world. Some of the features in the system needed modification, particularly when applied to problems with discrete variables and multiple objectives. The ability to view individual details of designs was a further suggestion by the contributors. The enhanced system was shown to other experts, instigating discussion and feedback that provided further insight into the potential of the system for knowledge discovery and evaluating the robustness of solutions.

1.3 Contributions to Knowledge and Results

The main contribution to knowledge is the development of a system that combines the research areas of engineering design, evolutionary computing and multivariate visualisation, enabling the understanding of problems in a novel and clear way. This is a new approach to understanding the output of data from a genetic algorithm that will clearly benefit many engineering design applications. Individual features of the system include:

- A fast, novel clustering algorithm based on kernel density estimation (Silverman 1986) that defines pertinent regions of the search space in terms the original variables or alternative coordinate systems such as the principal components
- A novel method to evaluate and confirm the robustness of regions using 'negative' GA search to find the worst case and a filtering mechanism to redefine clusters
- A mutation scheme that prevents the duplication of individuals in the GA
- Data can be visualised and analysed in 'natural' coordinate systems; clusters are highlighted with colour so they can be related to the original design variables
- A flexible interface that allows GA search and clustering in any coordinate system, inside or outside any region defined by the user or clustering algorithm
- The ability to focus on parts of objective space using penalty functions, generating new information and designs that are most relevant to real world problems

The artificial multidimensional test functions used during the user evaluation experiments are also a novel addition; they were designed to simulate problems found in engineering design. It could not be statistically proved that the users performed better than the benchmark algorithms on the engineering design task because the paradigms were more successful on some aspects of the problem. However the system received positive feedback from the users and it was shown that even a devoted algorithm would need help to solve the engineering design task. The critical analysis of the system suggested some modifications to the clustering procedure that would help to ensure all the required information is made obvious to the user. The analysis also suggested a simpler, more practical routine to help assess the robustness of solutions, whilst concluding that any routine needs to be adapted to the engineering design task presented. Overall the potential of the system is confirmed and suggestions for future testing are given.

The system was developed for continuous problems and then tested on a variety of engineering design problems. The studies confirmed that the system provides a valuable addition to the engineering design community, promoting knowledge discovery and understanding in many real world problems. Visualising the trade-off between objectives and using the system to generate new solutions in pertinent regions of objective space provided 'better' solutions to previously published results. However the true strength of the system is its ability to support decision making by supplying a diverse range of alternative design options, particularly when the constraints on the problem are not explicitly given and the engineer is contemplating many design scenarios. Viewing the individual details of designs (or a representation of the artefacts) helps the engineer assess their relevance to the design problem. The clustering and robustness evaluation procedures were less useful in discrete domains, but it was suggested that the user should be given more control to choose the variables or objectives used in the analysis. Furthermore it was shown that the system could theoretically be used to evaluate the feasibility robustness of solutions.

Feedback from the engineering design experts revealed that a lot of design knowledge is encoded into practical problems, but decision making and robustness evaluation is performed subjectively using experience of the problem. The difference between the theoretical concepts applied in the user evaluation experiments and the practical knowledge used in real world design scenarios was therefore highlighted. Further work was suggested, such as modifying the system for discrete and combinatorial problems, assessing the application of visualising and clustering data in alternative coordinate systems and the inclusion of advanced multiobjective optimisation techniques.

1.4 Overview of the Thesis

In Chapter 2 the literature in evolutionary computing applied to engineering design is critically reviewed and the need for the system proposed here is outlined. An extensive literature review of multivariate visualisation techniques, paying particular attention to the visualisation and analysis of engineering design data, is given in Chapter 3. The clustering technique that is tailored to produce information relevant to engineering design is presented in Chapter 4; the method is based on univariate kernel density estimation and can be applied in any coordinate system. Chapter 5 describes the system that was developed by the author based on the visualisation literature, enabling users to apply genetic algorithms flexibly to engineering design problems. Chapter 6 introduces the artificial objective functions and engineering design task created to test the system and the methodology used to compare the performance of novice users and benchmark algorithms. Chapter 7 presents the interesting results of the testing exercise, including a critical analysis that suggests modifications to the system, further ways to evaluate the robustness of solutions and procedures for future experiments. The case studies of experienced engineers evaluating the system working on real world problems are given in Chapter 8; analysis of the results and feedback provided by the engineers are given for each study. Finally conclusions and suggestions on how to exploit the potential shown by the system are given in Chapter 9.

Chapter 2: Interactive Evolutionary Computation applied to Engineering Design

2.1 Introduction

Many researchers have attempted to capture the design process using evolution and other techniques. This chapter begins by describing a number of definitions of design, demonstrating that the process depends on the people doing the designing as well as the object or entity that is being designed. The review concentrates on engineering design and the importance of finding robust solutions to the problem. With the advent of computers it is usually possible to generate a number of solutions to an engineering design problem and allow the designer to use their knowledge and creativity to verify those solutions or try other parameter settings to find more robust solutions. In a multidimensional problem this process can be limited if the user cannot visualise the solutions or indeed see how the solutions are related to each other. Visualisation of the design space would immediately improve understanding of the problem and allow the engineer to assess the robustness of solutions.

There are a number of ways the data can be generated even before visualising the search space, from random sampling to customised optimisation algorithms. Evolutionary computation is one of the most efficient ways of *exploring* a search space whilst *exploiting* the good solutions found, thus potentially robust solutions may be generated in this way. One particular form of evolutionary computing, the genetic algorithm, is suited to many engineering design problems and is briefly described along with extensions intended to ensure diverse solutions will be produced. The field of interactive evolutionary computing is particularly active and relevant to engineering design, so the relevant literature is also reported in this chapter. It was found that while a number of systems have been successfully developed for specific engineering design problems, there was no

evolutionary architecture that promotes visualisation and understanding of a general engineering design problem. The issue of robustness is also rarely catered for. Visualisation of evolutionary computation is also reviewed and was mostly found to be limited to understanding the evolutionary process; analysis of this information may lead to more efficient ways of producing robust solutions to a problem, but is not immediately relevant to the practical engineer. The conclusion of this chapter summarises the requirements of an interactive evolutionary system to support practical engineering design and describes the further research and processes required to build the system.

2.2 Engineering Design

2.2.1 Introduction

The process of design is iterative (Eckert *et al.* 1999) following a repeated cycle such as: formulate the problem, create the design, evaluate the design, reformulate the problem and so on. This is true for any sort of decision making process, from software development (Boehm 1988) to the completion of a large engineering project. There are many different versions of the design process that can be broadly categorised as either systematic or creative design, although Cross (2000) argues that a rational, systematic approach can lead to creative designs. In this section the different design methodologies are reviewed and compared to discover how the user and computer can work together to produce robust and creative designs and how evolutionary computing can support this goal.

2.2.2 Systematic and Creative Design

All designing is determined by a perceived need to make a product, demanding the formulation of a product specification or design brief. A typical design process by French (1999) is summarised in Figure 2.1. Engineering design is focussed on the manufacture of the product. A typical engineering design process called the Total Design activity model by Pugh (1990) is summarised in Figure 2.2. Both models shown in Figures 2.1 and 2.2 have a

loop that allows backtracking to previous stages due to the lessons learnt during the design process. These are descriptive approaches that concentrate on generating a solution early on in the design process and adapting the solution mostly by trial and error using heuristics built up by knowledge of the problem. This method can often lead to unfeasible design solutions and the need to begin the loop again (Cross 2000, p. 30).

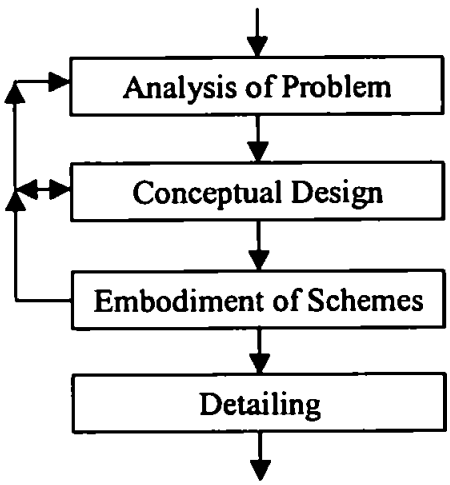


Figure 2.1:
Design Process (French 1999).

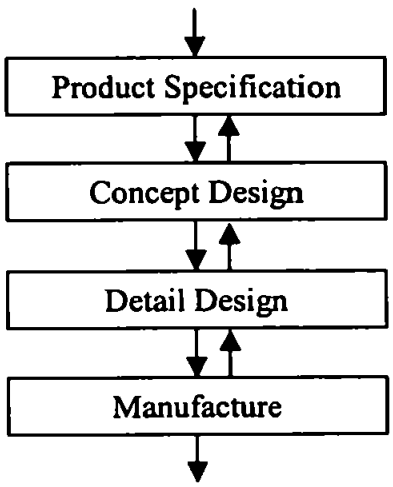


Figure 2.2:
Engineering Design (Pugh 1990).

To reduce some of the error in descriptive design some engineers prefer the systematic approach to design, summarised by Jones (1992, pp. 49-50). This is a prescriptive method that is almost algorithmic, encouraging rational thinking and analysis within each design stage. The iterative steps are: analysis, synthesis and evaluation, making informed decisions between the steps so that a number of well thought out design concepts are produced. Pahl & Beitz (1996) proposed a complete engineering design system based on this systematic approach. It could be argued that this method constrains the designer from creativity since the steps in the process are planned (Jones 1992), however Cross (2000) asserts that such a systematic procedure can be used to widen the original design space with the safety net of knowing that good quality design decisions have been made (*ibid.* p. 56).

After studying all the major views Dym (1994, p. 17) defines engineering design thus:

> Engineering design is the systematic, intelligent generation of specifications for artifacts whose form and function achieve stated objectives and satisfy specified constraints.

Because the aim of engineering design is to manufacture a product, it is important that the design meets the original specifications and is robust to manufacturing and material discrepancies. Discovering a flaw or inconsistency, not identified during the conceptual design process, forces the loop to begin again. It is important to make the right decisions at the preliminary and conceptual stage to avoid making costly mistakes; thus consideration of the robustness of the design is required as reviewed in Section 2.2.4.

Creative designing is often supported by dialogue between a number of designers; in particular discussing ideas is essential for a large scale design project (Bucciarelli 1984, Baird *et al.* 2000). Putting thoughts into words is an effective method for teaching design (Schön 1984, 1988), allowing the student to investigate new ideas and the teacher to impart experienced knowledge. Analysis of the conversation between novice and experienced engineers reveals firstly that a great deal of implicit knowledge is used by experienced engineers to make faster design decisions (Schön 1984) and secondly that designers have very different perspectives on what is a good design (Schön 1988). This reflective practice reveals how designers attempt to resolve conflicts and suggests that the ambiguity in design should be recognised, allowing participants to negotiate and eventually improve the quality of the design (Bucciarelli 1988). Study of these social and ethnographic processes usually involves painstaking analysis over many years that often provide a deeper understanding of design. For example, Baird *et al.* (2000) identified how specialist

engineers hunt for information from generalist engineers; they defined ethnographic terms to understand this interaction between engineers that will enlighten and assist future studies. However Bucciarelli (1984) admits that a purely ethnographic analysis may result in a design vocabulary that is too rich to be useful and may bias future designing.

As well as considering dialogue during the design process, Schön & Wiggins (1992) emphasise the importance of the materials used and how their visual interpretation will impact on the outcome. They describe designing as reflection and action, in other words 'seeing' the design in terms of perceived quality or visual gestalts, 'moving' to a new design, then 'seeing' again. This requires redrawing the design many times, possibly labelling the individual parts and explaining the moves to another designer (although as an engineer becomes more experienced the explanations will become less necessary). The next section describes attempts to model this process using computers.

In practice each model of the design process can and should be used in conjunction with other models, so systematic design can be used to produce creative solutions and the social impact on design should be considered whilst the engineering codes are being consulted, as concluded by Bucciarelli (1984).

2.2.3 Computer Models of the Design Process

Preliminary and conceptual design needs a lot of guidance by a designer or design team. Computerisation of the task would allow many designs to be evaluated before they need to be manufactured. Capturing preliminary or conceptual design in a computer program is a difficult task (Pham & Yang 1993a), although there have been numerous attempts. Some have tried to give a formal definition of the design process so that it can be used to make a computer design. The Knowledge Level by Newell (1980) proposed that an agent could undertake designing by giving it goals and rational behaviour to achieve those goals.

Smithers (2000) developed and tested such a method that formulates the knowledge needed at each step of design with partially successful results. Maher (1990) was concerned that using the prescriptive approach to model design is restrictive as they tell designers what to do, but not how to do it. She suggested that computer based models should facilitate the designer rather than prescribe design.

Other methods of capturing the design process include Gero's "situated learning" (Reffat & Gero 2000), that assumes knowledge is more useful when learned in the context of the design situation, and "design rationale" developed by Brown (Burge & Brown 2000) that captures the decisions and the reasons behind each decision during design. These models are attempts to recreate the moves and analyse the explanations between design stages, as described by Schön & Wiggins (1992). A lot of heuristics and learning methods are used to simulate this view of design, but it is doubtful whether the full complexity of the design process can be simulated in this way.

Coyne (1990) puts forward the seemingly contrary view to the cognitive models described above; this is the connectionist view that design can be modelled without explicit explanations and labels. If one models how the brain learns (at a very general and simplistic level) using design variables as inputs and the resulting designs as outputs, then examples of those designs can be retrieved by exposing various inputs to the trained network (see Coyne *et al.* (1993) for details of this process). However new (or imaginary) designs can be retrieved just as easily if unusual inputs are put into the system; these may be valid answers to the design problem. Hence Coyne (1990) argues that a model of design does not necessarily have to follow the traditional cognitive route of drawings and explanations to succeed. The model has drawbacks, such as how to represent a complicated design scenario for the system to learn. Also the black box nature of the result is a cause for concern; without an explanation of the output it will be difficult to justify the design.

Sim and Duffy (2000) performed protocol analysis on a designer at work to evaluate a model of learning in design. Methods such as protocol analysis usually involve the activities of a single engineer that was not found to be useful for large, multidisciplinary design teams (Jagodzinski *et al.* 2000); in this case social interaction has a large impact on the design process so the ever changing needs of the customers and company managers, as well as the morale of the design engineers, should be taken into account. This socio-technical approach underlines the need to consider the human perspective when organising design activity (Mumford 2003).

Because designing is a cognitive activity, attempts to capture the design process is the same as asking: *What does a person do when they are designing?* The answer to this is different for every person and problem, depending on the amount of knowledge and experience of the problem the person has (Eysenck & Keane 2000, Ch. 14). Gero (1990) gives three classifications of design: routine, innovative and creative. *Routine* design is constrained by pre-defined regulations, variable ranges and parameter values. A design activity is *innovative* if the variable ranges can be changed so designs can be found outside the expected domain. *Creative* designs are found by changing variable parameters and adding new variables to the model (Gero & Kumar 1993). As an example, Hoffman *et al.* (2001) describe an urban planning environment that benefited from a designer having the capability to turn all the rooms on the first floor through 90 degrees; this design was outside the initial constraints of the problem. However Maimon & Horowitz (1999) claim that creative designs must be in the 'neighbourhood' of the original design specifications, otherwise the problem has changed. The jury is still out on the exact definition of creative design, but it is encouraged by giving designers time to think about the problem, trying different scenarios, adding new information and possibly changing the problem (Eysenck & Keane 2000, Ch. 15).

Many of these models of design are attempting to describe an abstract process that is difficult to represent in a computer. Many researchers from the ethnographic community recommend that a computer system should provide an environment that facilitates design rather than attempting to model the complete design process (Bucciarelli 1988, Baird *et al.* 2000, Schön & Wiggins 1992). Researchers from the artificial intelligence community may argue that modelling the design process can facilitate design (Maher 1990, Gero & Maher 1993), but a different way to accomplish this is described in Section 2.2.5.

2.2.4 Robust Design

An additional aspect of engineering design is the need to produce robust design solutions to the problem, that is to try to find parameter or variable settings so that small or large changes in the settings cause small changes in the quality or cost of the product. Genichi Taguchi and his students undertook an in-depth study of robustness and quality (Taguchi 1986, Phadke 1989). Because of this work the difference between the quality of products produced in the USA and Japan became evident during the 1960s and 1970s. The American philosophy was to use known materials for an engineering task and specify tight tolerances on that material to produce a product of desired quality (known as tolerance design). Taguchi's philosophy (known as parameter design) was to assume low grade materials were to be used and to set wide tolerances on the noise and control factors (or parameter ranges), then try to find parameter settings that define a design within specification (Phadke 1989, pp. 33-34). To reduce the cost of a product (including the operating cost due to failure in the hands of a customer, known as quality loss), parameter design should always be performed before tolerance design. Taguchi (1986) performs parameter design by undertaking a small number of experiments defined using orthogonal arrays; parameters are tested a number of times and the outcome suggests a sensitivity ranking of each parameter due to a particular design objective. Analysis of these results can lead to a set of parameters that are robust in terms of a design objective.

According to Du & Chen (2000) most practitioners define robustness as "optimising the mean performance" and "minimising the performance variance" of an objective. Additionally the most popular way to evaluate robustness is through the worst case analysis concept (see Parkinson *et al.* 1993 for example). This procedure leads to conservative and sometimes infeasible designs, but because of the lack of guidelines and understanding of the problem it is the technique most often used. The importance of undertaking sensitivity analysis was stressed by Savic & Walters (1996) in their criticism of a method for designing water distribution networks by Eiger *et al.* (1994). In general this task is very difficult because of the size of the search space and the many different factors that cause sensitivity, there are also a number of alternative procedures that could be used to analyse the sensitivity (de Schaetzen *et al.* 2000).

Tweedie *et al.* (1996b) demonstrated how tolerance design can be visualised by specifying the tolerances in 'parameter space' (this is design or variable space) and using colour coding to highlight designs that satisfy the tolerances and also satisfy performance criteria (objective space). Colours vary depending on the number of parameter or performance limits that are satisfied. The authors also define the 'yield' of solutions that are acceptable due to performance criteria and can be manufactured according to the tolerance limits; very tight tolerances will result in a yield of 100%, but will probably be very expensive while wide tolerances may reduce cost but could result in a lower yield (some solutions will not satisfy performance criteria). This method depends on the random generation of solutions inside a default set of limits; if the user specifies tight tolerance or performance limits, the specified region may not contain many solutions so the system, developed by Tweedie *et al.* (1996), will generate more random solutions in this region. Random sampling, however, cannot guarantee that all solutions inside the region will behave in the same way, more intensive investigation may be necessary.

2.2.5 Summary: Why use Evolutionary Computation?

Rather than applying some abstract design model to capture the human designer's preferences (Gero & Maher 1993), an alternative approach (to find an innovative or creative design) is to generate a number of feasible or near-feasible solutions to the design problem and let the designer choose which solutions are best suited to the purpose and represent the best compromise between design objectives. In this way design is facilitated by the computer, instead of driven by the computer it in an attempt simulate the design process. Providing many solutions will enable the designer to reflect on the problem (Schön & Wiggins 1992) and make new design decisions, either in a systematic or creative way (Bucciarelli 1984).

The vast amount of computational power now available means that if a design objective can be modelled or approximated using mathematics then the simulation time on a computer is usually reasonable. Evolutionary algorithms have the ability to explore and identify good solutions in a pre-defined search space without *a priori* knowledge. Indeed Bentley (1999) postulates that computers can evolve creative designs, but humans are needed to identify the novel and useful designs from the infeasible and bad designs. In a conceptual design system the human can assess the quality of solutions found and guide the system to extend exploration outside the current search space. The search and exploitation characteristics of evolutionary computation can also be used to identify particularly bad solutions inside a region that can improve confidence in the evaluation of robustness of the region.

The following section gives a brief description of evolutionary computation (and the genetic algorithm in particular) and discusses methods from the literature that attempt to maximise the diversity of solutions returned by the algorithms.

2.3 Evolutionary Computation

2.3.1 Introduction

Evolutionary computing is the generic term for a group of techniques that were inspired by Darwin's theory that variation and natural selection is necessary for a species to adapt and succeed in its environment (Darwin 1859). The genetic algorithm (GA) is one form of evolutionary computation developed by Holland (1975) that includes genetic theory by representing a problem as a binary string and using crossover to pass on inherited information from generation to generation. Variation or mutation of bits in the binary string allows the evolving population to search outside the constraints of the previous generations. Alternative paradigms such as evolution strategies or evolutionary programming (reviewed by Bäck & Schwefel 1993) were developed at about the same time using a real valued representation. The paradigms are similar but vary in the implementation and the relative importance of each operator. Evolutionary algorithms have an advantage over many other optimisation algorithms because they do not concentrate search locally but combine information from multiple individuals in a parallel search and are more likely to find other optima through the action of mutation.

2.3.2 The Genetic Algorithm

Goldberg (1989) popularised the genetic algorithm and emphasised how its search and optimisation capabilities have been applied to many research and industrial techniques, in particular biology, psychology, pattern recognition and engineering applications. Much of the popularity and success of the genetic algorithm is due to its relative simplicity and robustness. The GA tends to perform well on many problems because it adapts to the problem domain itself; the only limitation is the difficulty in finding an appropriate representation and the definition of 'good solutions' for the GA to use. For engineering problems that involve discrete parameters in particular, the GA operators may return infeasible solutions unless repair operators or domain knowledge is included in the

representation (Rafiq & Southcombe 1998). Real valued GAs, evolution strategies or evolutionary programming may be more suitable for specific problems that require integer or floating point representation (Savic & Walters 1994). The canonical GA using binary representation will be described here because it is the type of evolutionary computation used in this research; it will be described primarily in the context of engineering design.

The genetic algorithm (GA) creates new solution to a problem following the iterative steps given in Figure 2.3. In contrast to other types of evolutionary algorithms each individual in the GA is represented as a set of genes called a chromosome in analogy with biological evolution. Each gene on the chromosome can take on one of a set of values, known as an allele. It is common for the chromosome to be represented as a binary string so the allele is either 0 or 1. The GA population is usually initialised randomly with alleles. Figure 2.4 illustrates how the binary string (or genotype) is decoded to the real-valued variable space (phenotype) for the engineering design problem. The variable space, also known as search or decision space, is usually bounded by limits given by the engineering problem or the designer. The variable values are in turn evaluated by an objective function to determine how good the solution is in terms of the engineering problem. It is possible that the engineering problem has more than one objective and they may be conflicting, so it is common to combine the objectives in some way to give a single fitness value.

Members of the population are then selected for reproduction a number of times depending on their fitness values and the selection technique used (Baker 1987). The selected individuals are paired off to generate children via crossover at the chromosome level (Figure 2.5). Additionally each gene (or bit) of the chromosome is subject to mutation at a low rate of probability (Figure 2.6). These operations occur in the genotypic space, so the binary representation is convenient, however the operators can be applied in a real-valued situation (such as simulated binary crossover and self-adaptive evolution strategies

(Deb & Beyer 1999)). The child population is then passed to the next generation and evolution continues until terminating conditions, such as convergence criteria or some pre-determined number of generations, have been met.

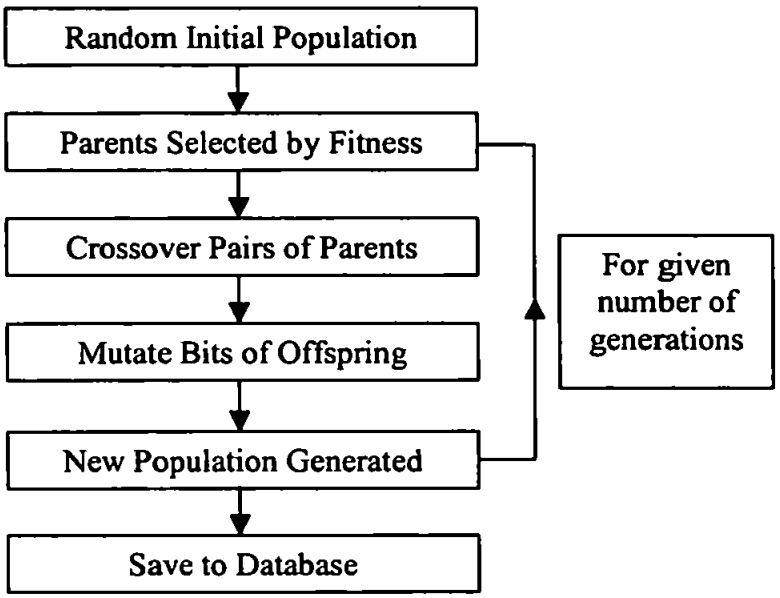


Figure 2.3: The Simple Genetic Algorithm.

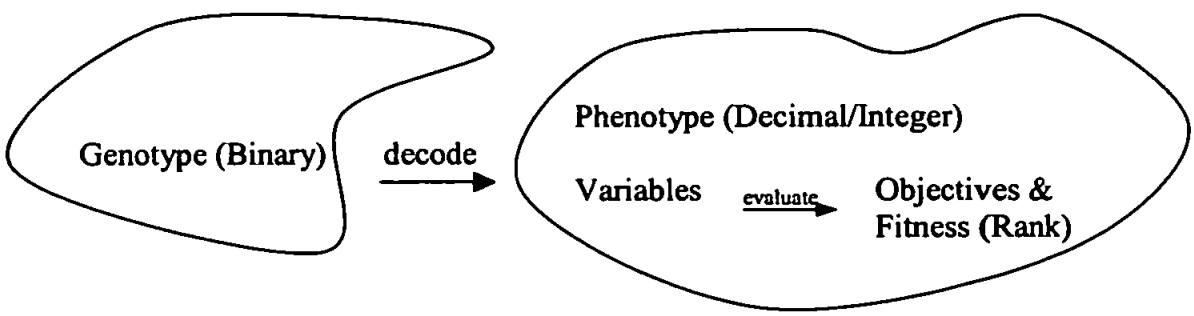


Figure 2.4: Mapping from genotypic space to phenotypic space; after evaluation the objective values and thus fitness rank is found.

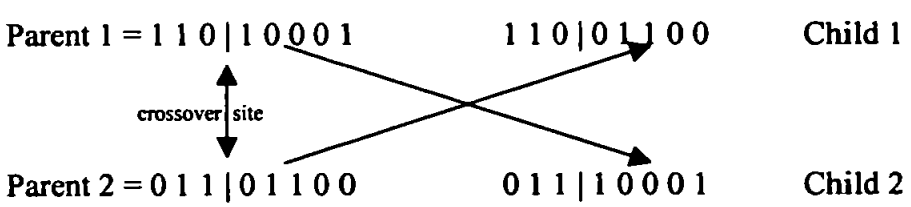


Figure 2.5: Crossover.

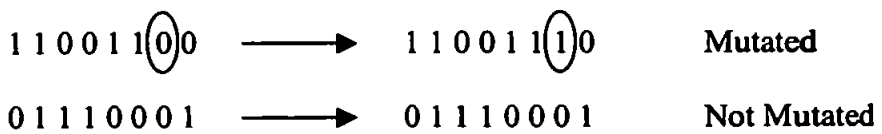


Figure 2.6: Mutation.

The selection, crossover and mutation operators will encourage the population to explore various options. In the simple GA good solutions will continue to be selected and the whole population will eventually converge on a local optimum in the fitness landscape (derived from the engineering objectives). Most practitioners then examine the final population or even just the best individual in that population, others identify good solutions to keep from each generation (Parmee 1996), but there is no reason why all the data generated by the GA cannot be saved and examined by the engineers. The latter strategy is adopted in this research.

The exact reason for the success of GAs is as yet unproven (Bentley 1999). They have been shown to be successful in many diverse problems and applications (Goldberg 1989) because the GA will nearly always return one or more regions of the search space that are worthy of further investigation and usually will return a large number of design alternatives in engineering problems.

There are two main drawbacks of the GA that are particularly apparent in functions containing multiple local optima (known as "multimodal" functions) and a large number of variables. The first problem occurs when too many diverse solutions exist in a population with similar fitness causing the GA to exhibit random search without getting a foothold on a good solution. Introducing elitism so that a proportion of the best individuals always survive in subsequent generations will help solve this problem; allowing the GA to concentrate search in specific areas. Conversely the GA may converge on a local sub-optimal solution where mutation is not enough to find more diverse solutions in the population without a long wait; techniques to slow down convergence are described in the following sub-section.

2.3.3 Maintaining Diversity through Multimodal Optimisation

The main requirement of an engineering design system is the ability to explore complex design spaces allowing the exploitation of known good regions of the search space and exploration of regions outside the known regions. In a conceptual design system one way to tackle this problem is by the interaction of human knowledge combined with the computer's data generation capabilities (Parmee *et al.* 2000). Evolutionary computation provides one of the most adaptable and explorative tools for data generation. But evolutionary algorithms will still stagnate to local optima without external help. This section briefly reviews multimodal techniques applied to evolutionary computation that could be used in interactive engineering design.

Researchers use multimodal optimisation techniques to maintain diversity in the GA by influencing either the objective space or decision space. The former approach is to modify the fitness function to allow search away from a converged region of the search space. The "sharing" algorithm (Deb & Goldberg 1989) encourages diversity by reducing the fitness of solutions according to their closeness to other solutions. Similarity between solutions can be determined by comparing solutions in genotypic space (the number of bits that are different in the binary string) or phenotypic space (the difference in the decoded variables, usually measured using Euclidean distance). The distance between all solutions is calculated, if the distance between a pair of solutions is less than a predetermined value (σ_{share}), then the fitness of each is reduced proportionally. Another parameter (α) controls the rate of fitness reduction. Deb & Goldberg (1989) give a method for determining the parameter σ_{share} by assuming that the peaks are equally spaced throughout the search space and that the number of peaks present is known or can be estimated.

The sequential niche technique (Beasley *et al.* 1993) searches for new peaks by "derating" the fitness of peaks already found, in a similar way to the sharing algorithm.

However this method suffers from the same problems as sharing (and the dynamic sharing method of Miller & Shaw (1996)): to be successful the peaks need to be evenly spaced and the sharing parameter needs to be carefully chosen otherwise the derating process can cause new artificial peaks in the search space or the algorithm will merely find the bottom of peaks already identified.

The second approach to multimodal optimisation influences the search in decision space by attempting to keep diverse solutions away from each other during replacement or crossover. One of the first attempts at controlling convergence was the "crowding" technique by De Jong (1975). Crowding uses a steady state GA such that a subpopulation (proportion G of the main population) is chosen for reproduction. Children of the subpopulation are compared with a certain number of the main population and replace the individual that they are most similar to (in genotypic space). The number of individuals used for comparison is called the "crowding factor" (*CF*). *CF* is usually set at 2 or 3 resulting in two or three niches being maintained in the population, this is known as replacement error and is a major criticism of crowding. Increasing *CF* may increase the number of maintained niches, but overall performance is reduced, eventually the whole population will converge anyway. The ecological (ECO) GA (Davidor 1991) also uses a steady state GA and encourages separate niches to flourish in a population by simulating reproduction among 'neighbouring' individuals. Again the population will eventually converge so some online metric or visualisation technique is required to observe interesting results from the ECO GA.

Crowding has fewer parameters than sharing, but Mahfoud (1992) introduced "deterministic crowding" (DC) that virtually eliminated the replacement errors of crowding and does not requires additional parameters to the simple GA. At every generation the whole population is paired off, crossover and mutation is applied to each pair of parents,

producing two children. Each child then competes with its most similar parent for survival and will replace the parent if it has a higher fitness. This algorithm allows a number of subpopulations to be maintained in the population in an efficient way. The similarity between individuals is performed in phenotypic space (as in phenotypic sharing), which is a great improvement over comparing in genotypic space (as in original crowding) because adjacent designs in binary space are often very far apart in variable space, even if grey coding is used. DC will distribute solutions proportional to the width (or hypervolume) of peaks in variable space, although if two local optima are adjacent the one of higher fitness will dominate and the lower peak will be lost (Mahfoud 1994).

Multimodal optimisation is particularly relevant to engineering design for which Parmee (1996) developed the Cluster-Oriented Genetic Algorithm (COGA), which samples all the information generated during search. This means that clusters of good solutions found during early stages of the GA search can be saved thus avoiding loss of information due to the convergence of the algorithm. Like sharing and crowding this technique requires parameters that need careful tuning and often *a priori* knowledge of the search space. Deb *et al.* (2000) developed the crowding technique without parameters to adapt to the current information and Pham and Yang (1993b) successfully applied various multimodal techniques to the design of their preliminary engineering design system. However in general it is difficult to find a technique that adapts to all possible fitness functions, thus in recent years evolutionary research has reverted to more fundamental theories of genetics and concentrates on attempting to find a smooth mapping between genotype and phenotype space to assist the evolutionary process (Harvey & Thompson 1997, Shipman *et al.* 2000).

Humans are good at inferring information from data, so combining the human and computer directly during a GA search can only improve the exploration. Rather than

guessing the number or width of optima before a search, it is proposed to let the user explore data generated by the GA and make decisions based on that exploration. The use of current multimodal techniques will obviously be beneficial but the parameters that control them need to be determined during the search. Most of the literature on multimodal optimisation uses either test functions with one or two dimensions, such as the sine function (Deb & Goldberg 1989), or multidimensional problems with regularly spaced peaks, such as De Jong's test suite (Goldberg 1989, pp. 108-110) and the functions proposed by Ackley (see Bäck & Schwefel 1993), Rastigin, Griewangk and Schwefel (given in Gordon & Whitley 1993). Engineers may have some idea of the number of peaks or at least the location of good solutions in problems they have defined, but will not appreciate making multiple parameter changes to locate those regions. Additionally when looking for innovative or novel solutions, engineers have less knowledge of the search space and may need to break out of their pre-conceived assumptions.

The next section critically reviews the literature to find a system that combines the human and evolutionary computation to effectively analyse and explore the design space. The main requirements of such a system is to identify robust regions within the known search space and support exploration to find novel solutions outside those already found.

2.4 Interactive Evolutionary Systems

2.4.1 Introduction

There are two main approaches to interactive evolution. Some researchers define interactive evolution (IE) or an interactive genetic algorithm to be a special type of evolutionary algorithm where a human user performs evaluation or selection thus subjectively capturing the designer's requirements (Graf & Banzhaf 1995, Takagi 2001, Caldwell & Johnston 1991). The alternative approach is to design an interactive system that uses information from an evolutionary process or genetic algorithm (Parmee *et al.*

2000, Pham & Yang 1993a, Jo 1998). These latter systems allow the user to try new variable ranges or manipulate the system in some way but do not insist that the user selects individuals for the next generation. In fact because of the large number of individuals used in these systems it is not viable for a human to evaluate all the designs that are generated (Hudson & Parmee 1995). The former type of interactive evolutionary computation (IEC), characterised by human evaluation or selection, is the 'narrow' definition of IEC according to Takagi (2001). While he gives the 'broad' definition of IEC to human-computer interactive systems that use evolutionary computation. In this thesis the term interactive evolution is used with the broader definition in mind unless qualified by the word narrow.

2.4.2 Narrow Interactive Evolutionary Computation

Narrow-IEC has been successfully applied to face recognition (Caldwell & Johnston 1991), generation of jazz music by responding to a human musician (Biles 2001), sound and computer aided design (CAD) applications (Takagi 1996) and engineering design (Graf 1996). Takagi (2001) gives a review of around 250 research papers on interactive evolutionary computation (mostly narrow). There are some interesting designs created using variations of narrow-IEC. Graf & Banzhaf (1995) used an expansion operator that applies regression to the input from the user to evolve graphical shapes of fish and cars. Sato & Hagiwara (2001) evolved novel cutting tools by combining different stages of the design process; handles and blades of the tool were evolved by human selection independently, the next stage was to combine the handle with the blade, in later stages the user refined the designs further.

The advantage of subjective human selection over traditional evolutionary computation is the use of knowledge or intuition of the user to evaluate designs (Herdy 1991). The user's choice and preferences are directly captured in the evolutionary system. The main drawback of narrow-IEC is the 'fitness bottleneck' (Biles 2001) which is related

to human fatigue (Takagi 2001). There is a limited number of evaluations a person can and will be motivated to perform in a certain time.

To overcome the problem of the fitness bottleneck Caldwell & Johnston (1991) introduced a number of techniques including optimising the starting position of individuals and changing crossover and mutation rates online. This application concerns the recognition of Photofit-like pictures by witnesses. The approach is interesting because it uses the human's powers of pattern matching rather than feature recall (humans find it difficult to describe another face but accurately recognise faces from an early age). The user is shown twenty faces and asked to rate the fitness of each face according to its similarity to the remembered suspect. Once the witness had found a close match they found it useful to fix the parameters of desired features of the face to prevent the loss of that information due to future mutations.

Takagi (1996) also applied narrow-IEC to drawing faces imagined by the human. To speed up the search process the members of each population were sorted in order of likely fitness. This fitness was derived from a neural network (NN) by extrapolating the fitness given to similar solutions in previous generations. The algorithm by Takagi (1996) shows that a computer can simulate the human through fitness evaluation, even though the human's evaluation is psychological and subject to change. Takagi points out that it is surprising that a NN can learn to generalise in such a large parameter space and hypothesises that this is because the NN is only learning a small part of the whole parameter space. But this emphasises the point that using human interaction in this way will generally focus the search and cause stagnation of the search process.

Indeed Biles (2001) found that using a human to evaluate Jazz tunes resulted in a certain style being returned that the musician (Biles himself) found boring. The solution to

this problem was to effectively remove fitness from the system. New populations were generated automatically using domain knowledge rules of jazz and intelligent crossover. The convergence problem was removed, but the fitness landscape is basically flat because crossover and mutation will always result in a jazz phrase that is 'safe' or musically plausible.

In general a knowledgeable engineer will build the variables and objectives in an engineering design system. But the aim of conceptual or preliminary design is to discover innovative or creative solutions to problems. This means dropping routine design and looking outside the search space of known solutions to the problem. In this case rather than using the user to select solutions for future generations, it is desirable for the human and computer to look in regions outside the normal search space. So an evolutionary system is required that can confirm the quality or robustness of solutions already found and allows the user to look for other good solutions. Such a system will reside in the broad view of IEC. That is evolutionary systems that are directed by a human user either by manipulating where the search takes place or influencing which objective has more importance, but the user does not actively select individuals for future generations. There follows a review of such systems.

2.4.3 Broad Interactive Evolutionary Computation

Pham & Yang (1993a) developed an often-cited preliminary design system called TRADES (TRAnsmission DESsigner) that incorporates a genetic algorithm (GA). The components of the system such as gears, shafts and belt wheels are the building blocks. The building blocks are quite large and further domain knowledge is present in the system because the designer chooses specific design requirements. The GA produces configurations for the user to evaluate; infeasible configurations are penalised using a simple fitness function. Many design options are available because the GA incorporates a

number of multimodal techniques (Pham & Yang 1993b). The engineer can view all the raw data from the GA or details of individual design configurations. So individual designs can be chosen or another GA run performed as required. This system is very domain specific as the components of the system and how they fit together are incorporated into the algorithm. New design solutions can be generated, but they are presented in a text-based interface so differences between solutions may be difficult to comprehend.

Jo (1998) discovered that adding human interaction to his evolutionary design system meant domain knowledge could be incorporated online. He acknowledges that (*ibid.* pp. 219-220):

> Many researchers have attempted to model human intelligence in the computer system, in order to make an intelligent design system. However, it may be an easier and better idea to include a human designer as a process module in the design system, rather than trying to build an intelligent computer program for design. By doing so the system will bring the designer into the design process actively and dynamically. Designers will then be able to reflect their ideas in the process quickly and efficiently. The design solution space will be expanded, shrunk or moved dynamically, as the designer's perceptions, interests or design requirements changes.

He allows the user to interact with the system at all stages of the GA process: initialisation, evaluation, selection and recombination. Solutions are independently visualised in a space layout problem and the user is allowed to modify individual elements of the design. The user can also give arbitrary fitness values to the designs. In this way the user can direct the search using design knowledge and intuition to guide the algorithm away from a local optimum. One of the main aims of adding the user to the system was to solve the convergence problem; the GA will quickly converge to a solution but it will take some time to improve the solution after convergence. So the user will attempt to find new solutions using intuition and domain knowledge. There is no indication that the system suggests the location of other good solutions.

A user's input greatly improved the speed of convergence and quality of solutions to the travelling salesperson problem according to Louis & Tang (1999). The complexity of this problem is exponential, so for a large number of cities the traditional GA takes a long time to find useful solutions. Using this method the user selects clusters of cities to 'divide and conquer' the problem into a number of smaller problems for the GA to work on. The user then reconnects the sub-problems. This example shows the great advantage of an interactive genetic algorithm, although the authors point out that the user's input may guide the search process to sub-optimal solutions as a lot of the search space will be ignored.

The interaction of a user has also been considered in a multiobjective environment. Fonseca & Fleming (1993) proposed a decision-maker (DM) that controls which objectives have more importance within a non-dominated set of solutions. They suggested the DM could be human or an expert system. The objectives were visualised using a parallel coordinate type of technique (see Section 3.3.5). The multiobjective genetic algorithm was put forward as a good way to bring decision making to engineering design. Balling *et al.* (2000) devised a multiobjective decision making tool to interactively create and analyse potential city plans with visual feedback. Horn (1997) points out that there are three different approaches to decision making in multicriteria problems: make a multicriteria decision *before* search, make a decision *after* search or *integrate* the search and decision making. The latter approach would appear to be the most powerful, incorporating iterative search and decision making. Mathews & Rafiq (1994) also suggested that adaptive search could by harnessed to assist a multidisciplinary design team in reaching a compromise between constraints and feasible solutions at the preliminary stage of structural design.

Ian Parmee proposed an Interactive Evolutionary Design System (IEDS) based on a system of iterative redefinition of variable and objective space by a designer as search progresses (Parmee & Bonham 1998, Parmee *et al.* 2000). Evolutionary modules generate results in a multimodal and multiobjective environment that can be evaluated off-line by the user. Identification of high performance regions is available due to the evolutionary process and confirmed by the user with visualisation techniques. Subsequent redefinition of variable limits or change in preference of objectives is supported. Except for the input of preferences exactly how the information between user and computer is exchanged is not discussed. The visualisation technique of individual two-dimensional plots is limiting and does not take into account the complex interactions of high dimensional variables (Packham 2000). However the tool is a good approach to conceptual design, it is intended to complement the human (Hudson & Parmee 1995) and allow the designer to discover more about the search space by exploiting what is already known and exploring other alternatives. The work on COGAs (Parmee 1996) that identified clusters of good solutions during the genetic algorithm run was extended with particular emphasis on the extraction of knowledge (Bonham & Parmee 1999); visualising the output of GA runs using different parameter settings improved understanding of multimodal functions and the location in decision space of the best compromise between multiple objectives.

2.4.4 Related work

There has been some interesting research in techniques that are not interactive evolutionary systems *per se* but are relevant to the discussion of how to involve the user in design.

Matthews *et al.* (2000) trained a neural network on patterns during an interactive design process to discover how the decision space is mapped onto evaluation space. Features were extracted using principal component analysis (PCA) and self-organising maps (SOMs) that described the design heuristics enabling the authors to develop design

guidelines and general concepts used to form an initial evaluation of the design process. This method shows the usefulness of PCA and SOMs to describe the interactions of high-dimensional data. The main components indicate which variables have most affect on important objectives and each other. Josephson *et al.* (1998) encourages exploration of large search spaces by filtering good solutions according to the non-dominance criterion. Good trade-off solutions are then displayed between two criteria at a time; the user can choose subsets of solutions from the interface for further examination. This technique is not evolutionary but allows the user to inspect and interact with generated designs.

John Gero has done a lot of work attempting to capture the design process in evolution, although without using interaction explicitly. In Gero (1998) he postulated that the building blocks of the genetic process contain the design information and used genetic engineering to exploit that information. This system thus evolves problem specific knowledge and due to the genetic engineering is likely to converge more quickly (Gero & Kazakov 2000), but may also stagnate at a local optimum without human intervention. Bentley (1999) also encodes domain knowledge into evolutionary systems and attempts to evolve new designs using the computer without human intervention. However in the conclusions of his thesis, Bentley (1996) suggests input from a user during the evolution of solid objects via an interface would facilitate the design process.

These examples indicate that PCA, SOMs and evolution can capture design heuristics and important design features. However a human is needed to interpret the features, choose between design options or help the algorithms escape from local optima. Therefore visualisation of the search or solution space and analysis of the regions discovered is essential in an engineering design environment. A review of the literature concerning visualisation applied to evolutionary computation follows to discover the state of the art research in this area to date.

2.5 Visualisation of Evolutionary Computation

There has been a lot of interest in the visualisation of evolutionary computation (EC) in recent years as evidenced by workshops devoted to the subject (GECCO 1999, Smith *et al.* 2002). This is motivated by a yearning to understand the underlying process and to see how much decision and problem space the GA has covered. In order to assist the user during engineering design using EC it is necessary to display the information generated by the GA so that the user can see the different sort of solutions that are generated, the robustness of those solutions and identify areas of the search space worthy of further investigation. Review of the literature reveals two main categories for EC visualisation research: visualisation of the ongoing EC process and visualisation of the search space covered by the algorithm (Hart & Ross 2001, Collins 1997, Shine & Eick 1997, Pohlheim 1999). In this section visualisation techniques applied to evolutionary computation will be summarised, more details of the algorithms and guidelines for visualisation will be given in Chapter 3.

Visualisation of the ongoing EC process involves looking at individual chromosomes or the fitness of solutions to see how the algorithm is progressing from generation to generation. Online visualisation of fitness versus generation graphs and genotypic or phenotypic information helps the user evaluate the efficiency and dynamics of the GA operators. Pohlheim (1999) devised a set of standard techniques for this type of GA visualisation. A sophisticated and recent technique visualises the course of the GA process (Hart & Ross 2000). They describe a new tool called GAVEL that attempts to help the user understand how the GA travels along a certain search path using colour and shading to represent fitness, phenotype and allele values. At the end of the run the best chromosome is found and its evolutionary history is inspected by backtracking. This enables the user to identify how different GA operators such as crossover and mutation have affected the search. This technique does not take into account multiple solutions or multimodal fitness

functions. How solutions evolve is interesting and can teach us how to optimise the GA parameters, but most users in an engineering design environment will not be GA experts. Engineers will be more interested in the quality (in terms of objective value) and robustness (in terms of the change in variables that provide the objective value) of solutions. In related work Ross & Corne (1994) point out how GAs can be most useful as simple hill climbers when applied to problems and too much time is wasted on tweaking GA parameters for minimal improvement.

The visualisation of decision and objective space is more applicable to engineering design applications. If the problem is multidimensional most researchers attempt to view the problem in fewer dimensions using principal component analysis (PCA) (Collins 1999). PCA enables the natural distribution of the data to be viewed from an orthogonal set of vectors called eigenvectors. The first two or three eigenvectors usually contain most of the information about the variability of the data. However the two main problems with multidimensional scaling (MDS) techniques such as PCA are: firstly the result is a distorted representation of the original and secondly there is no consistent spatial relationship between successive views of the principal components (PCs), a particular problem as data generated by the GA is always changing (Collins 1997, Hart and Ross 2000). Collins (1999) went some way towards rectifying the second problem by visualising the GA progress using a manifold of PCs over generations enabling the user to assess the efficiency of the GA and determine when it is stuck at a local optimum.

Most MDS methods try to represent the high-dimensional search space on a low-dimensional picture whilst retaining the spatial distance between points. This technique means finding the optimal representation in low dimensions that correctly represents the real distances in high dimensions. Pohlheim (1999) used SAMMON mapping (Sammon 1969) to do just this; optimising the starting configuration of the algorithm using PCA.

However the algorithm is very computationally expensive and prohibitive for a large number of data points. Shine and Eick (1997) also represented high dimensional data in two dimensions using a similar method to SAMMON mapping, although it does not preserve distances from generation to generation. Additionally they used quad-codes to estimate how much solution space has been explored, but this technique does not retain the relative distance between individuals. Collins (1997) suggested an interesting technique called 'search space matrices' that maps every chromosome onto a two-dimensional space using the allele values. The distance between chromosomes is preserved. However in engineering design the interest is in the difference between variables of solutions as well as the overall difference between solutions.

In their review of the state of the art in visualisation techniques for GAs, Hart and Ross (2001) have a small section on visualising the whole space searched by the GA. Those few researchers that have attempted this have tried to represent the whole space in two or three dimensions and find it difficult to assess the effect of GA parameters on the search or differentiate between landscapes. Again an engineer will be less interested in how the GA is performing, more in what parts of the search space have been explored and how different variables interact. This type of question leads onto more general visualisation of high-dimensional data.

Spears (1999) reviews other multidimensional visualisation tools that have been applied to GA visualisation. Pictures can be used to represent the data in a form that is readily recognised by a human; these are sometimes called icons or glyphs. For example Chernoff faces can represent values of different variables by the size of features on a face. This technique takes advantage of the facial recognition ability by humans. There are a limited number of variables that can be represented on the face before the complexity makes even these difficult to understand. Projection techniques are also mentioned as a

way to represent high-dimensional data in low dimensions. The most interesting projections indicate the main clusters in the data and which variables contribute to those clusters, however these algorithms are optimisation techniques that attempt to find certain shapes of data; looking for 'interesting projections' may take a long time and often need human supervision (see Section 3.4.5 and Cook *et al.* (1993)).

The problem with most high-dimensional visualisation techniques is that in order to be understandable by a human the representation is distorted to be viewed in lower dimensions. This means some of the actual information is not preserved in the transformation, in particular subtle changes in the data and individual variable information will be lost. There is one visualisation technique that preserves the original information and can be viewed on a two-dimensional picture. Parallel coordinates display all variables and objectives as vertical lines. Individual points in Cartesian space are represented as lines in parallel coordinates between the variables. This means the mathematical structure of the original data is preserved (Spears 1999). Fonseca & Fleming (1993) showed results of multiobjective interaction using parallel coordinates (they did not refer to them as such). By temporarily constraining the limits of one of the objectives and redisplaying they could look at individual solutions in more detail.

In this section visualisation techniques applied to EC have been reviewed. It would appear that the needs for visualisation of EC applied to engineering design have not been catered for. Rather than visualising genetic information and how chromosomes have evolved from previous generations, the requirements for engineering design are the visualisation of individual variable information as well as the overall distance between solutions themselves. So there is a need to develop an interface where the distance between regions of solutions is obvious in terms of each variable. Also interaction should be supported so the user can easily change the ranges of the variables and zoom in on

different portions of the search space. Two visualisation techniques appear to support this: one option is to use parallel coordinates, the other is to represent all the data in a number of two or three-dimensional graphs and allow the user to compare variables with each other. The second method is more familiar to engineers and the GA users who often see data represented as a fitness landscape.

2.6 Conclusions

The use of interactive evolutionary computation in engineering design is in a healthy shape according to various writers. In their study of the applications of genetic algorithms (GAs), Ross & Corne (1994) claim that GAs have been much more accepted than, say, expert systems. GAs are easier to use and more understandable and they can find quick solutions to problems without endless tweaking of parameters. Takagi (2001) unsurprisingly predicts that evolutionary computation will become very important in design and creation systems, making the computer more user-friendly. However there is a large gap to be filled between the potential use of evolutionary computing in engineering design and actual systems that flexibly allow users to reap the benefits. Most decision support tools that use genetic algorithms are designed to solve specific problems such as building design (Mathews *et al.* 1996), water system design (Morley *et al.* 2001) and mechanical engineering (Pham & Yang 1993a), but a system that can represent all problems in an easy to understand way and allow the easy implementation of genetic algorithms in the problem would broaden the scope of applications that could be used and increase the likelihood of engineers trying out the new technology. However the complexity and diversity of engineering design problems are such that some tailoring will be required for any system, as a potential user needs to become comfortable with the system as well as be an expert in their problem.

Missing from current interactive evolutionary systems is the ability to understand the solutions in variable space and the capability of the human to interact with the search

space and guide the system towards innovative solutions. Such a system should give clues as to the location of other high performing regions, if possible. Caldwell & Johnston (1991) allowed the user to fix parameters during the evolution of suspect faces for recognition. This type of interaction is useful to exploit certain attributes of search, but the opposite is also needed where variables are set free to find new solutions. Therefore engineering design and evolutionary computing can both be enhanced with appropriate visualisation tools to help understand the search space allowing interaction by the user to guide the search. Current visualisation research in evolutionary computing does not cater for this need.

In a forward thinking appraisal, Lund (2000) compares two interaction styles: *direct manipulation* of the traditional graphical user interface and *interactive evolution*. The direct manipulation style provides control to the user with little surprise while interactive evolution can be used with little training and may provide a lot of surprise and add to the creative process. He claims that the combination of the two styles can complement each other in supporting creativity. To include direct manipulation in an interactive engineering design system based on evolutionary computation (interactive evolution in the broad sense) a review of the general information visualisation literature is needed. A comprehensive overview of all related techniques is given in Chapter 3.

Evaluating robustness in engineering design is a difficult issue mainly because the definition of robustness is problem dependent. In this thesis, robustness is defined with the assumption that engineering data will be generated using evolutionary computing. At the local level a solution is defined as robust if changes in variable values cause little or no change in objective values. Hence a set of solutions is defined as robust if they are neighbours in variable space and their corresponding objective values are similar (see Section 6.3); these sets of solutions are termed 'robust regions' of the search space. To find

these regions, the philosophy of Taguchi (1986) points to setting wide tolerances in variable space and attempting to find solutions that satisfy various engineering requirements (Section 2.2.4). Tweedie *et al.* (1996b) provided a visualisation system (described in Section 3.6) that allows the examination of feasible solutions in variable space with tolerances chosen by the user superimposed on the plots. The technique was limited by the random generation of solutions (as was another similar proposal by Josephson *et al.* (1998) that used dominance filtering to present good trade-off solutions to the engineer), but if such an idea is extended to include optimisation techniques such as the genetic algorithm then the engineer can have more confidence that the solutions inside a region are the best that can be found. Equally in order to test robustness of the region the conservative 'worst case scenario' concept can be implemented using 'negative' optimisation, that is using the GA to find the worst solution in a region. A system that includes such additions is described in Chapter 5 and is novel, at least in regard to the application of evolutionary computing to engineering design.

To evaluate the robustness of engineering design data it is necessary to identify the interesting clusters with respect to the problem domain, that is the objectives used by the GA. Techniques such as density estimation, clustering and PCA can speed up some of the data analysis tasks (Matthews *et al.* 2000) and explain how variables interact, but the results need to be presented in a way understandable to the user and relevant to the design process. Mathematical and clustering techniques are reviewed in Chapter 3 with particular emphasis on identifying robust regions returned by a genetic algorithm.

An interactive evolutionary system will allow the user to freely explore the search space supporting innovation and possible creativity. To test such a system it is necessary to develop test functions with multimodality in high dimensions containing an unknown number of peaks and random placement of peaks. This is in contrast to most of the

literature on multimodal optimisation that either use test functions with peaks that are evenly spaced and it is known beforehand how many peaks there are in the search space (Beasley *et al.* 1993) or multidimensional problems that are designed to be difficult for GAs but the global optimum can be deduced by visualisation or inductive search (Bilchev & Parmee 1996). When applying their multimodal optimiser to an engineering problem many researchers report a few results that have no meaning to the reader so the success cannot be substantiated (for example Roy *et al.* (1996)). Therefore test functions with engineering design features and location of peaks that can be analytically defined but difficult to solve using a genetic algorithm are presented in Chapter 6.

After the review of the general visualisation literature and associated techniques given in Chapter 3, a novel clustering technique used to locate regions pertinent to engineering design is presented in Chapter 4. The proposed interactive system using evolutionary computations will be described in Chapter 5. This system is motivated by some of the concepts and ideas that followed from the development of COGAs (Bonham & Parmee 1999) and the Interactive Evolutionary Design System (IEDS) proposed by Parmee *et al.* (2000). The novel additions to the system described in this section bring the proposal some steps towards reality and at the very least offer another useful decision support tool enabling knowledge discovery as suggested by Mathews & Rafiq (1994) and Mathews *et al.* (1996). It is hypothesised that the interactive visualisation system will support systematic design (Cross 2000), reflective design (Schön & Wiggins 1992) and social or multidisciplinary design (Jagodzinski *et al.* 2000). Whether the system can be extended to capturing the design process and provide truly creative designs, as suggested by Gero (1990) and Gero & Kumar (1993), is up for debate. A lot of research in this area (Gero & Maher 1993) would suggest that implementing such a creative design system is possible.

Chapter 3: Visualisation for Interactive Engineering Design

3.1 Introduction

The power of human perception and cognition can identify clusters in data that statistics find difficult to describe. However the same powers can suggest fictitious clusters or imaginary relationships in the data because of the assumptions made by the human. Therefore careful design and implementation of an interactive system is extremely important to ensure that the human can interpret information in the correct way. The first part of this chapter will investigate some of the theories behind human computer interaction and general visualisation guidelines. Particularly relevant to engineering design are multivariate and multidimensional visualisation techniques, which are described in Section 3.3 using the classic Iris Data as an example.

The aim of this investigation is to find a set of visualisation tools relevant to interactive engineering design and evolutionary computation. Pure visualisation tools rely heavily on the user who may not have the time or inclination to perform a complete analysis. The advantage of using a human is that they may see unusual patterns of data or use their intuition to search, but the user can also get stuck looking in regions of the search space that are not important. Conversely statistical and clustering methods can be perceived as a black box that simply tell the user where to look next; this leaves the engineer asking why is the information important and where do the results come from. So a balance is required between using pure visualisation or human perception and statistical or clustering techniques. A broad range of statistical analysis and clustering methods are described in Sections 3.4 and 3.5. Then state of the art multivariate visualisation systems that have been used or can be applied to engineering design are investigated before conclusions are drawn at the end of the chapter.

3.2 Human Computer Interaction Guidelines

3.2.1 Introduction

Building a successful interaction between the human and the computer is fundamental to a controllable visualisation system. Before designing the interface a number of issues need to be researched and addressed. A basic understanding of the psychological processes being used during human computer interaction needs to be understood. In this section a brief summary of the psychological literature is given followed by an overview of human computer interaction guidelines, as well as recommendations regarding the use of colour.

3.2.2 Psychology

There are a large number of models attempting to describe what is happening in the human brain during interaction with a computer. For example the model human processor (Card *et al.* 1983): information passes through the perceptual system to perceptual memory, if it lasts long enough in perceptual memory the information will be represented symbolically in working memory. Once the perceptual processor has encoded the information, the cognitive processor will decide what to do with the information and tell the motor system to act accordingly. High-level theories such as the GOMS (goals, operators, methods and selection rules) model by Card *et al.* (1983) attempt to describe interaction in terms of the model human processor and predict how well a human will perform a specific task. Each task is broken down into a series of sub-tasks needed to achieve the goals using cognitive or motor operators. A number of methods may achieve the same goal so selection rules are used to choose between those methods. This theory can be used to validate the effectiveness of interfaces, however it does not suggest how to build the interface in the first place.

Norman (1986) introduces a theory of action that is similar to the GOMS model (Card *et al.* 1983), but emphasises the difference between psychological and physical

variables involved in human-computer interaction. He suggests ways to bridge the gulf between the user and computer by either bringing the user nearer to the system or the system nearer to the user. In this human centred approach the needs and interests of the user are stressed; it should be obvious what an object on the interface does by its physical interpretation. In later research he defines these clues to how an object operates as "affordance" (Norman 1998, pp. 9-10). A physical example of this is the simple question of how to open a door; a plate on the door implies it should be pushed to open, a bar implies it should be pulled. Unfortunately many doors are not designed in this way.

More recently Shneiderman (1998, pp. 61-67) presented the object-action interface model. This model describes direct manipulation interfaces that use objects such as buttons and dropdown menus. Most of the components of the system are already provided by object-oriented or GUI design systems, so the task is more to match the available objects to the set of required tasks. Some of the most important considerations are to promote consistency between programs and systems, so that the same key presses (CTRL-C for example) mean the same thing in all instances (usually copy). Preece *et al.* (1994) go further by declaring that traditional cognition does not go far enough; research on distributed cognition is needed to take into account the environment of the user and interaction with other users.

Some psychological experiments have supplied results that can be used in practice. For example Card *et al.* (1983) used Fitt's Law to show that the mouse had a clear advantage over other pointing devices on many criteria. Fitt's Law states that the time taken to point to an object on the screen is related to the distance from the target and inversely related to the width of the target. So the further away and smaller the target, the longer time it will take to point to the target. Additionally the time taken to react to visual feedback is approximately a quarter of a second, which is an eternity in terms of the speed

of a computer processor, so it is not necessary to update information any faster than this. In fact in some cases where decisions need to be made an update of 1 second may be acceptable (Spence 2003). The response time needed to connect events (that is to make a dynamic visualisation process look continuous) is 0.1 second (Card *et al.* 1999b).

Most researchers have come to realise that the only way to proceed with interface design is to pass experience in the form of principles and guidelines onto others so that mistakes are not repeated. This is because most interfaces are designed by trial and error with only passing reference to psychological literature. A lot of testing has also revealed the best techniques to use; some of these are given in Section 3.6.

3.2.3 General guidelines

One of the most useful and popular guidelines to interface design is Shneiderman's mantra:

Overview first, zoom and filter, then details on demand.

These are some of the tasks that need to be supported for all types of visualisation according to Shneiderman (1998, pp. 522-541). The other tasks that need to be supported are: *Relate*, *History* and *Extract*. So the idea is to get an *overview* of the data, then allow the user to *zoom* in on interesting items and *filter* out uninteresting items. If requested *details* of individual or a group items should be readily available and visualising the *relationships* between items will help understanding of the data. Keeping a *history* of actions will support undo or redo commands and allow users to change and refine their searches. *Extraction* is the process of saving interesting parts of the data and parameter settings so other users can discover how that data was arrived at.

Shneiderman (1998, pp. 67-79) also indicates three main principles of design, which can be refined and adapted for individual requirements and systems. The first

principle is to recognise the diversity of users; there are novice users, knowledgeable (intermittent) users and expert users. With a new system only the designers will be expert users so they should take into account the inexperience of other users. The second principle is actually eight golden rules of interface design, they are: strive for consistency, enable shortcuts for expert users, supply feedback about what the system is doing, design dialogues to yield closure, error prevention and handling, easy reversal of actions, make the users initiators of actions (so there should be no surprises, on the other hand the user should not have to perform many routine, tedious actions) and reduce short-term memory load (the rule of thumb is 7±2 chunks of memory (Miller 1956)). The third principle of interface design is to prevent errors. While it is useful to supply strong and unambiguous error messages after errors have occurred, Shneiderman (1998, pp. 76-79) suggests that it would be better to prevent errors happening before they can affect the system. Habitual users may ignore important instructions such as hitting OK to a save without quitting command with disastrous results (Garfinkel 2000, Raskin 2000); an alternative design may be required to prevent this error. Other considerations such as providing documentation and help files for the design system, keeping the display uncluttered and simple to reduce information overload and improve usability are common sense proposals.

While Shneiderman (1998) provides many useful guidelines for the design of interfaces, Hutchins *et al.* (1986) were concerned that providing an interface that is purely based on direct manipulation may restrict the actions that can actually be performed. It may be obvious what a button-click or drop-down menu will do, but does it reflect the requirements of the user? They suggest that the system should be designed for "direct engagement" so that the "user experiences direct interaction with the objects in a domain" (*ibid.* p. 114), in other words the user should be able to integrate with the system to perform the actions they want rather than be restricted by the objects on the interface.

Robert Spence of the Intelligent and Interactive Systems Group, Imperial College, London, UK has built many tools over the years using concepts and philosophies that were devised before computers were fast enough to draw multidimensional images and compute large data sets. These tools are all based on an effective combination of techniques described in Section 3.3; his interesting web page includes views on the past, present and future of interactive visualisation systems (Spence 2003), see also his book (Spence 2001). Initially Spence invented MINNIE, a tool for circuit design. He claims that success came from allowing invention on the part of the user by removing constraints and letting the user formulate problems as they search for solutions. The tool should facilitate exploration and not require unnecessary expertise to use. Input and output parameters should be on the same view so that the effect of parameters is directly seen. A simple menu was designed to navigate interactively and default behaviour was implemented to return the system to a familiar state. MINNIE was one of the earliest examples of direct manipulation before the term was coined. He also points out that focus and context is needed, but both of these cannot always be provided at the same time. For document viewing a bifocal display or fisheye view was used so that most of search space is in outline apart from the region of interest.

When considering automated design Spence (2003) points out that algorithms can make new designs, but the question is whether designers readily accept them. Do designers have the time or motivation to learn the new skills required to understand the algorithm or the new design produced? Additionally he found that icons did not give a significant advantage over text for selection tasks once the user had learnt the meaning of the symbols, especially when the user is choosing between a large number of parameters. These findings led to the development of Attribute Explorer and Influence Explorer described in Section 3.6.

3.2.4 Colour

One of the most controversial and difficult choices to make when designing a user interface is the number and selection of colours to make available to the user. The main problems are to mix colours that can be viewed easily without overloading the display. Colour should be used carefully to illustrate variation in important information because colour deficient viewers may not be able to see the difference and if the picture is printed off in grey scale the differences in the original colour may not be apparent.

Shneiderman (1998, pp. 398-403) suggests designing for monochrome first and adding colour to enhance the display. Tufte (1990) also recommends using saturated colours to emphasise important, unusual data while unimportant (close to average) data should be depicted as a grey or neutral background. A good example of this is seen in Carr (1994) in which the residuals of a statistical study on male colon cancer are shown on a map of the United States. Most of the map is grey indicating the average across the region; those regions with a noticeable deviation from the mean are shown in a light colour (red denoting higher than average, blue lower than average). Regions that are markedly different from the mean are depicted in a dark red or blue. Because these regions are minimal they stand out on the map and overall trends can be seen and related to local living conditions.

The combination of colours on a display can be critical in designing something that is easily readable and not too confusing. For example saturated red and blue next to each other will strain the eyes because they are on opposite ends of the light spectrum. For text or objects to stand out one should try to use colours that have very different brightness, for example black on white or white on dark blue. Figure 3.1 shows how different colours can be difficult to see depending on the background. Most colours can be seen against a grey background.

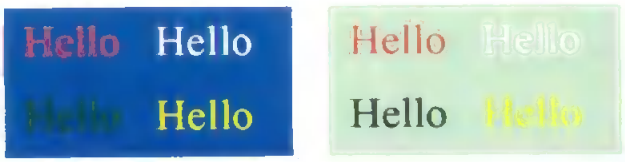


Figure 3.1: The effect of colour and brightness combinations on a display. In the left hand picture the white and yellow letters are easy to see whilst red and black are difficult to see on a dark background. The opposite is the case in the right hand picture on a pale green background.

Figure 3.1 shows another characteristic: a colour will not seem as bright in all conditions, depending on the brightness and wavelength of surrounding colours. This phenomenon is due to fundamental laws of the theory of light, such as Weber's Law: the brighter the stimuli are the larger the difference is needed between stimuli to make a noticeable difference. So for example the relative brightness of red and blue next to each other may change depending on the overall light conditions (Kaiser 1996). In general some colours are bad to mix and some can sit together nicely (Preece *et al.* 1994, p. 92) because of the difference in contrast between colours as well as the problems for users with a colour deficiency; a significant percentage of the male population cannot differentiate between red and green. However most people can perceive the difference in brightness and different shades in a grey-scale version of an image because of the way light is interpreted.

When light is emitted from the computer screen or reflected from a piece of paper, the eye and brain interprets the wavelength and intensity of the photons as a colour. One of the most accepted definitions of colour appearance is the hue, saturation and value model (HSV) that is used to select colours for painting (Padgham & Saunders 1975, p. 95-99). The HSV model is based on the psychological definitions of colour perception (Wyszecki & Stiles 1967, p. 229, Padgham & Saunders 1975, pp. 61-77). The 'hue' of the colour is the pure form of one of the spectral colours seen when white light is split by a prism. The

'saturation' measures the purity of the colour; a colour is made less pure with the addition of white light causing the colour to pass through pastel shades to a grey or white that is the least saturated. The 'value' is the brightness or luminosity of the colour which is the intensity of light being perceived, from black (no light) to very bright colours. All these factors suggest an infinite number of colours and shades are available but only a finite number of colours can comfortably be used in a display (Shneiderman 1998, p. 399). Figure 3.2a-c illustrates the HSV model; these 'colorbars' were generated in MATLAB that also provides a direct mapping between the HSV and RGB (red-green-blue) models.

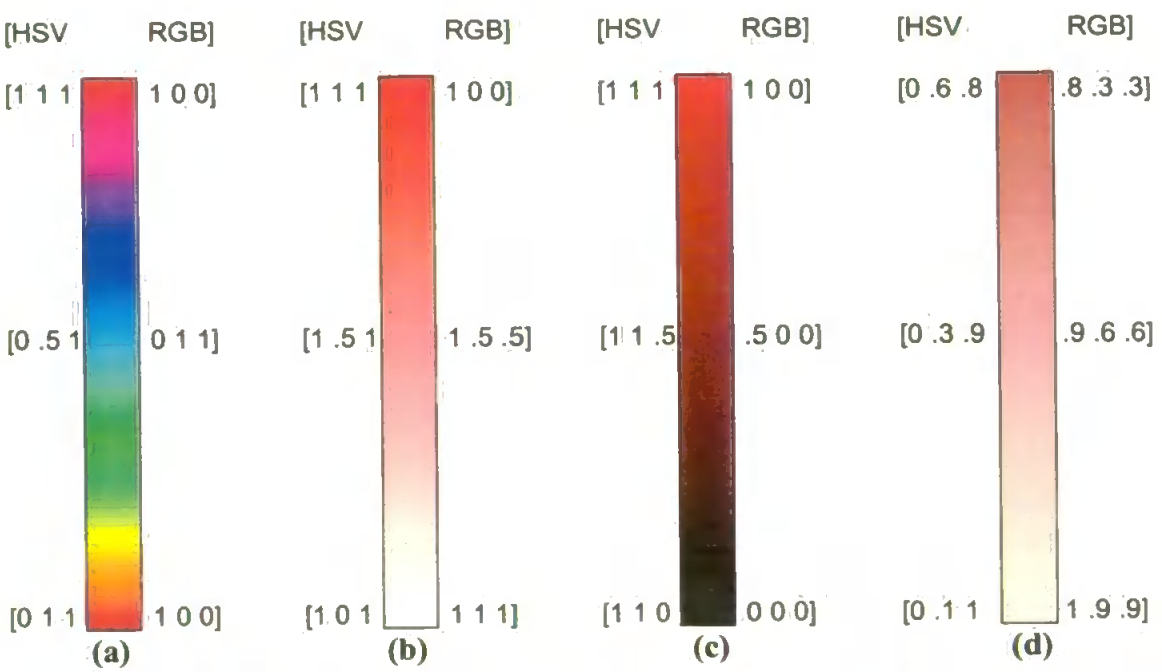


Figure 3.2: The HSV colour model illustrated by changing one attribute at a time: (a) hue, (b) saturation and (c) value. Changing saturation and brightness at the same time provides a large range of variation (d). The corresponding RGB values are also given. For both models each attribute is scaled between zero and one.

A visualisation system for engineering design requires the representation of both continuous attributes (to signify the change in objective value, for example) and discrete attributes (such as data with the same or similar variable values). The hue of colour will obviously provide a number of discrete cues while the brightness value (or luminosity) can suggest continuity. Saturation can be used to provide more discrete choices or more variability in the continuous attribute. Figure 3.2d shows the range of shades available

when brightness and saturation is varied for a single hue, the variation should be apparent to most people including those with a colour deficiency. Such a representation should be used to indicate change in the most important attribute of a display.

In summary colour can definitely enhance a display but can be confusing if over used. A number of (sensible) default colours should be provided but users will have their own favourite colour preferences so the ability to choose colours should be available, particularly for those users with a colour deficiency.

3.2.5 Visualisation Guidelines

Tufte (1983) suggests general guidelines on visualisation. The important lesson is to keep the amount of information that is not actually data (non-data ink) off the picture. For example gridlines and text describing the data is usually redundant or can dominate the actual data. The main principles during the graphical design process are:

- Above all else show the data
- Maximise data-ink ratio
- Minimise non-data-ink ratio
- Erase redundant data-ink
- Revise and Edit

Tufte also gives guidelines for graphical excellence suggesting that the best designs involve complex, often multivariate data presented with clarity and precision, so that the greatest number of ideas is communicated using the least ink in the smallest space. It is also important not to distort the truth about data using disproportionately sized graphics, a technique often employed by the media to misrepresent facts. Too much information on the same page can be confusing especially if multicoloured graphics are employed.

Particularly good designs are those that are drawn in a way directly relevant to the actual data, for example on p. 50 of Tufte (1983), the varying size of white pine plants due to varying amounts of calcium are depicted by the size of actual drawings of the plants. The use of colour in a sparing but meaningful way can enhance the graphic according to Tufte (1990). A small amount of saturated colour will emphasise the important information, this is seen in the design of traffic lights. The best design strategies reveal detail and complexity whilst leaving background information in a greyish or neutral colour.

3.2.6 Summary

Human computer interaction guidelines suggest that the best interfaces allow users to do what they like, within reason; with the ability to have an overview of the data, zoom and filter the data, then inspect individual data points (Shneiderman 1998, p. 523). The goal is to bring the user and system together so that physical objects on the interface relate to the psychological needs of the user (Norman 1986). This is not necessarily achieved by an interface that is purely based on direct manipulation, Hutchins *et al.* (1986) suggest some flexibility in the design would allow the user to become immersed in the system.

The literature suggests novice users would benefit from a simple button click interface, while knowledgeable users prefer more commands (perhaps on a menu driven interface). Common sense ideas to keep in mind are to maintain consistency of the system, design dialogues to yield closure, think about error handling and letting the user initiate actions. Where possible automating the system takes away tedious actions, but the user should be allowed to take over if they want to try something different. Back tracking to undo what the system or user has done is also very important. The user should be able to set the colour of regions, however default settings and colours should be used to give the user some feeling of familiarity. The interests of colour deficient users should be kept in mind when selecting colours to use; in most cases brightness of colour is the best attribute

to use to represent change in continuous data whilst the hue can highlight discrete attributes or groups of data.

Graphical visualisation can reveal complexities in data that simple statistical analysis may not, as demonstrated by Anscombe's quartet (Anscombe 1973). Figure 3.3 shows four sets of data that are described by the same linear model: the mean of x and y are the same, the equation of regression line, standard error of slope and correlation coefficients are the same as well as the root mean square in x and residual sum of squares in y. However the data pairs all look different to the naked eye. The difference in relations between the data becomes apparent once analysis of residuals and non-linear data modelling is applied to the data. In higher dimensions the human cannot see even these relations so more sophisticated statistical analysis is necessary to help understand the data. In general it is not sufficient to describe data using a linear model. Some relation between the variables needs to be computed and checked. To visualise results of this analysis multivariate visualisation and clustering is required as described in the following sections.

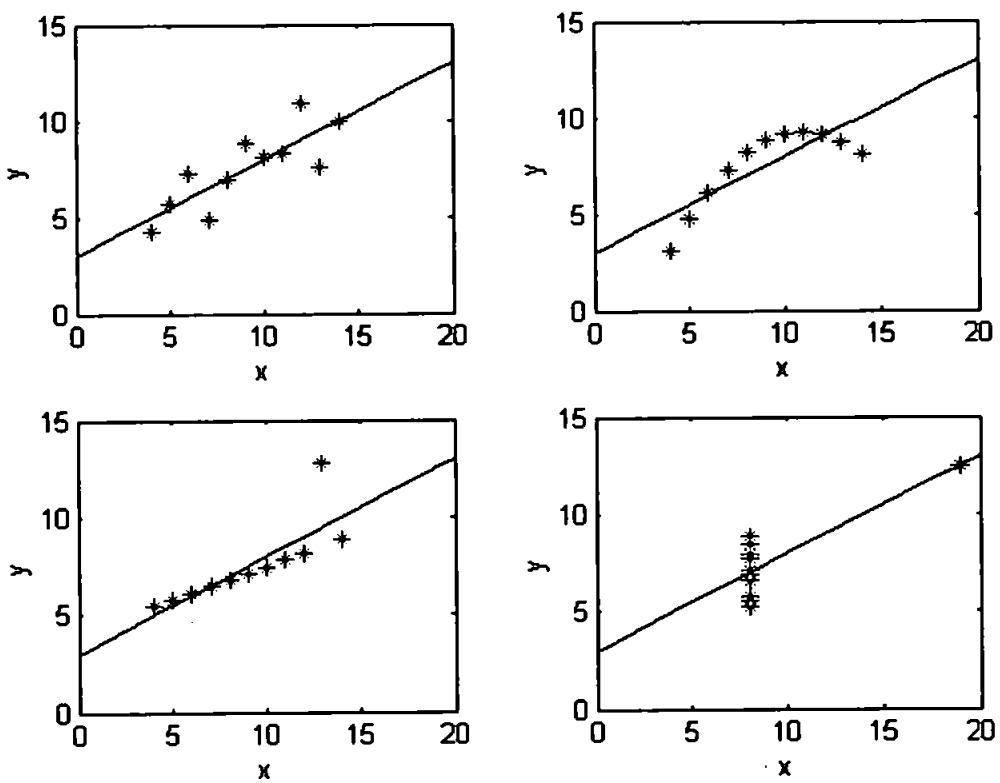


Figure 3.3: Anscombe's quartet, the same linear model describes all four data sets.

3.3 Multidimensional and Multivariate Visualisation

3.3.1 Introduction and the Iris Data

Many of the multidimensional visualisation ideas of today are inspired by the satirical book "Flatland: A Romance of Many Dimensions" by Abbott (1884). This book is written from the point of view of a square living in a two-dimensional world who encounters first some inhabitants of Lineland (1 dimension) and then the inhabitants of Spaceland (three dimensions). After much mathematical and physical persuasion the square eventually believes in the existence of worlds containing a different number of dimensions to Flatland. He then deduces the existence of Extra-Solids, that is shapes of four dimensions and the possibility of worlds with an infinite number of dimensions. From our point of view it is difficult to think of a world containing more than three dimensions, but we need to if we wish to understand general engineering and mathematical problems.

Wong & Bergeron (1997) provide an overview of 30 years of scientific visualisation. During the development of computers, researchers were forced to display data on paper or using simple displays. However the groundwork for the best data analysis techniques grew out of this era. *Exploratory Data Analysis* written by Tukey (1977) introduced new ways of thinking about decoding information from data. This started by writing down numbers in such a way that reveals the relationships between data (called stem and leaf displays, similar to histograms), then introducing scatterplot displays and regression lines to understand relationships between variables, to alternative displays such as boxplots and understanding the importance of residuals. As computers got faster so more data could be visualised and statistics computed faster, but since the 1970s only a few completely novel high dimensional visualisation techniques have been invented.

In their review Wong & Bergeron (1997) provide a distinction between *multidimensional* and *multivariate* variables. They define the dimensionality of the space

as the number of independent variables that describe the problem while the number of variates is the number of dependent variables. Unfortunately other researchers do not use the same conventions, for example Gilbert *et al.* (2000) use the term *multivariate* to describe raw data and *multidimensional* to describe data where only the distance between objects is defined. To avoid confusion in this thesis the problems will be explicitly defined in term of the dependent and independent variables. In general the term multivariate will apply to real–valued data sets, while multidimensional will refer to those problems where only the distance between objects is known. For example multidimensional scaling (MDS) is a form of multidimensional analysis that approximates the Euclidean distances between data points using a metric that can be more easily visualised in fewer dimensions.

During this discussion most of the visualisation examples will use a classic data set. The Iris Data is a well-known benchmark data set used to test pattern recognition and clustering algorithms. The data was collected by Anderson (1935) and made famous by Fisher (1936) when he used it in statistical analysis. Three varieties of the Iris flower are classified: *Iris Setosa*, *Iris Versicolor* and *Iris Virginica*. The Iris database consists of 50 examples of each class of flower, containing four variable measurements: sepal width, sepal length, petal width and petal length (see Appendix A for the actual data values). The *Iris Setosa* class is linearly separable from the rest of the data (seen most clearly in the lower ranges of petal width and length of Figure A.1 in Appendix A, see also Figure 3.4), but the *Iris Versicolor* and *Iris Virginica* are not linearly separable.

3.3.2 The Scatterplot Matrix: Additions and Variants

The subject of multivariate visualisation is categorised into three areas by Wong & Bergeron (1997): techniques based on two-variate displays, multivariate displays and animation. The two-variate displays generally consist of the scatterplot matrix (sometimes called the generalised draftsman display (Chambers *et al.* 1983), see Figure 3.4). The data

is projected onto two-dimensional plots and each variable is plotted against the others (see also Section 5.2.1, Figure 5.1). This display can be enhanced by adding fitting lines or important statistical details such as the mean, median or standard deviation. Figure 3.5 shows one of the scatter plots from Figure 3.4 with the linear regression line; box plots (or box-and-whisker plots (Tukey 1977)) of each variable are also shown. The box plots show the median value of the data, the upper and lower quartiles and the upper and lower adjacent values (Chambers *et al.* 1983, pp. 11-24). The interquartile range is the difference between the upper and lower quartiles and describes the spread of the solutions around the median. The upper and lower adjacent values are determined by the adding or subtracting the interquartile range (multiplied by 1.5) to the upper or lower quartile respectively. Outliers can be identified by showing the points that are outside the upper and lower adjacent values (in Figure 3.5 the outliers are depicted by red stars above and below the box plot).

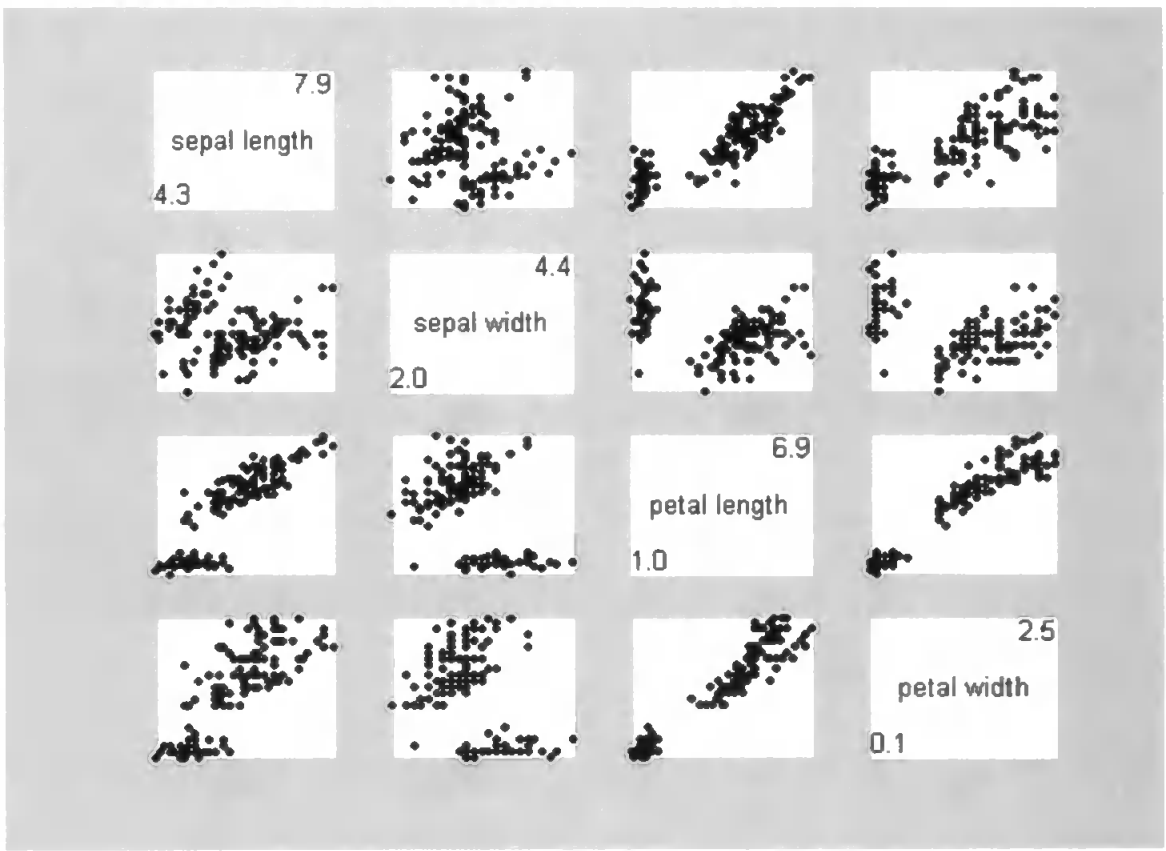


Figure 3.4: Scatterplot matrix of the four dimensional Iris Data. The windows on the main diagonal give the variable name corresponding to that row and column, the numbers in the corners are the minimum and maximum of that variable.

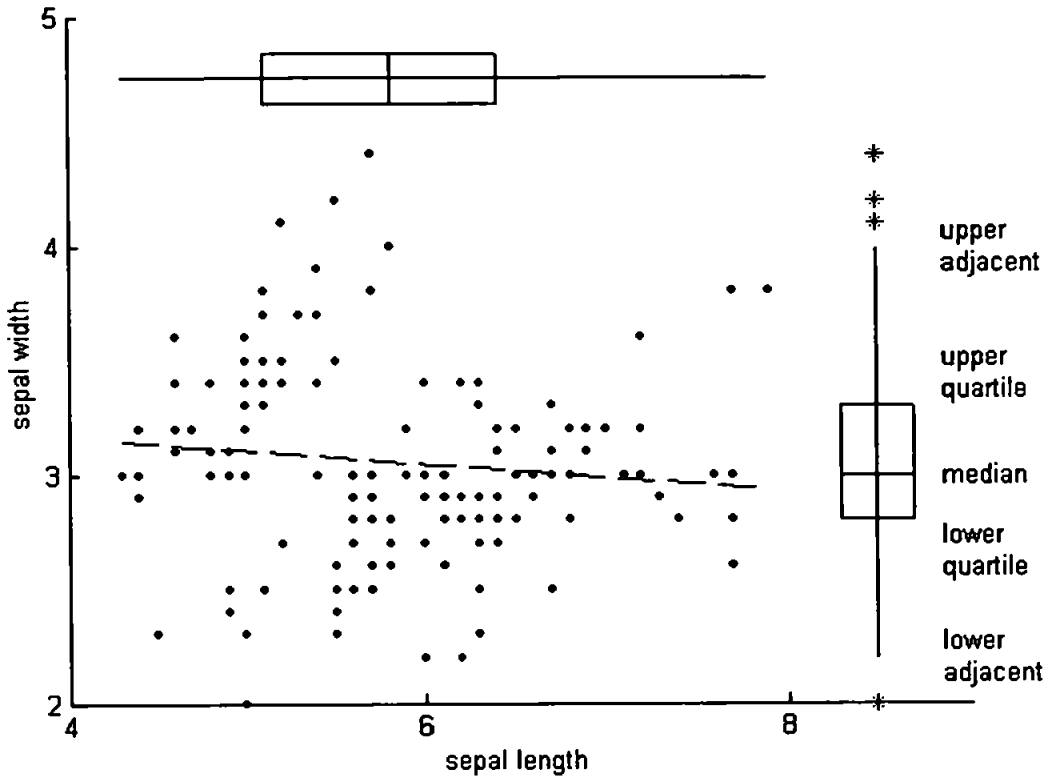


Figure 3.5: Box plots give a summary of the data in each dimension, the red stars are outliers. The dashed line is the linear regression line.

Often a linear model does not best describe the data. Chambers *et al.* (1983) describe a non-linear smoothing technique called lowess (or loess) to fit the data to a line. More sophisticated statistical techniques include analysis of variance and covariance, principal component analysis (PCA, linear) and independent component analysis (ICA, not necessarily orthogonal). All these techniques need confirmation by analysing outliers, residuals or hypothesis testing. Kohonen's self-organising maps (Kohonen 1997), *k*-means and hierarchical clustering also help to understand relationships in data; these methods are discussed later in this chapter.

One of the most common and useful techniques to help understand multivariate displays is brushing (Becker & Cleveland 1987). It allows the user to link a set of points and see the effect on all the variables. Martin & Ward (1995) developed a multidimensional brush for the XmdvTool (Ward 1994). The brushing tool can be used in

all the displays generated by XmdvTool: scatterplot matrix, glyphs (Section 3.3.3), parallel coordinates (Section 3.3.5) and hierarchical clustering (Section 3.5.2) displays. Figure 3.6 shows the multidimensional interactive brushing tool being used on the classic Iris Data (see Chambers *et al.* (1983) and Cleveland (1993) for extensive discussion and references on this data). The *Iris Setosa* subset was selected in the petal width versus petal height plot and the relevant points are highlighted in red in the other graphs. The brushing tool is very useful in helping the user identify clusters in the data and to understand the relationship between variables by visualising the effect of moving the brush. The fact that the same information can be displayed in a variety of different formats is also an advantage (Figures 3.4, 3.5, 3.6, 3.7 and 3.11).

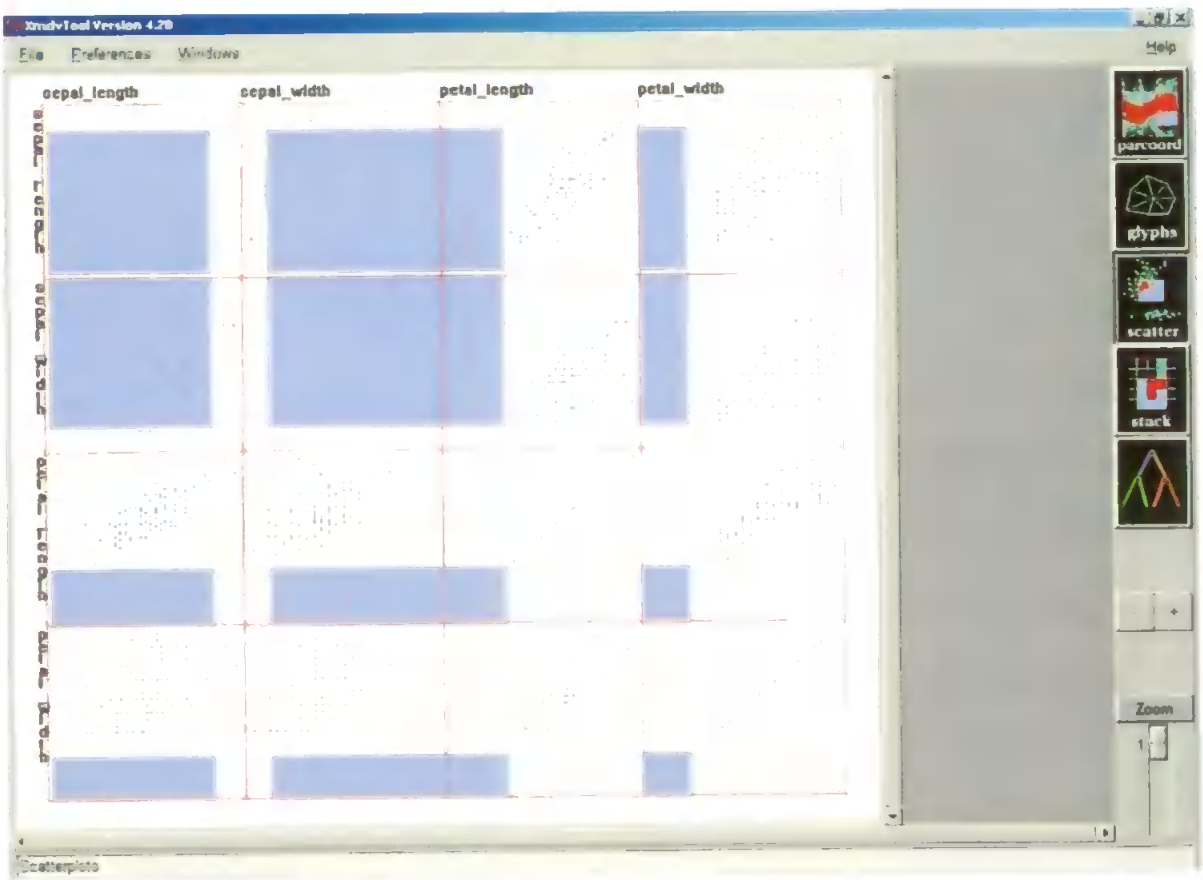


Figure 3.6: Scatterplots by XmdvTool (Ward 1994) showing the Iris Data, the interactive brushing tool highlights the *Setosa* cluster in the petal width and petal length variables. Used with permission.

Hyperslice by van Wijk & van Liere (1993) is another enhancement of the scatterplot matrix. A focal point of interest is defined by a centre point and a width in each dimension to create a region the user wishes to concentrate on. This technique allows direct manipulation by navigation of the focal point around the space. Hyperslice can also be used on discrete data. Various authors have used distortion techniques to concentrate on specific data in a two dimensional display by magnifying the data at the focal point or reducing the size of surrounding data using a variety of magnifying functions (Leung & Apperley 1994), the ideas have been extended to three dimensions by Carpendale *et al.* (1997). This kind of controllable distortion has an advantage over simply 'zooming in' as details of the data can be seen in the context of surrounding data.

Wong & Bergeron (1997) also mention a technique by Alpern & Carter (1991) that uses the information from a scatterplot matrix but displays them all as a side of a Hyperbox. Some of the sides of the box will be occluded by the others so an array of Hyperboxes is displayed all sliced at different places so the subsets of the data are seen. When drawn on the page or screen the sides of the box are all of different lengths so the data is scaled by a different factor in each variable. The Hyperbox is an example of the main stumbling block to visualisation using non-scatter techniques: the data needs to be interpreted as not all the actual information can be displayed if more than two dimensions are used. Some data will be occluded or distorted at different rates in each variable and the actual position of the data can be uncertain (this latter point is also true in three-dimensional scatter displays). With more than three dimensions some imagination is required to make sense out of the displays. In some cases it can take a long time to decode the information that is being presented. Apart from the scatterplot matrix, visualisation techniques tend to represent high dimensional data using abstract or iconic displays.

3.3.3 Iconic / Abstract Displays

Symbolic pictures such as icons, glyphs, stick figures and cartoon faces have all been used to display each data point individually. The value of each variable is represented by the size or shape of a component of the picture. Figure 3.7 shows a star glyph display used by XmdvTool. The length of each line emanating from the central point defines the size of that particular variable. Chernoff faces are a popular multidimensional display that depicts each dimension using a physical attribute of the face (Figure 3.8). The idea of this display is to take advantage of a human's face recognition ability. Each point is represented as a single face. As the number of variables increases the complexity of the faces will increase and understanding the relationship between variables is difficult. Also as the number of points increases the size of each glyph or face will become smaller to fit on the screen so comparing data is difficult. In general there is no natural order for the position of the pictures as each variable will change at different rates so finding clusters by comparing glyphs or faces will be difficult.

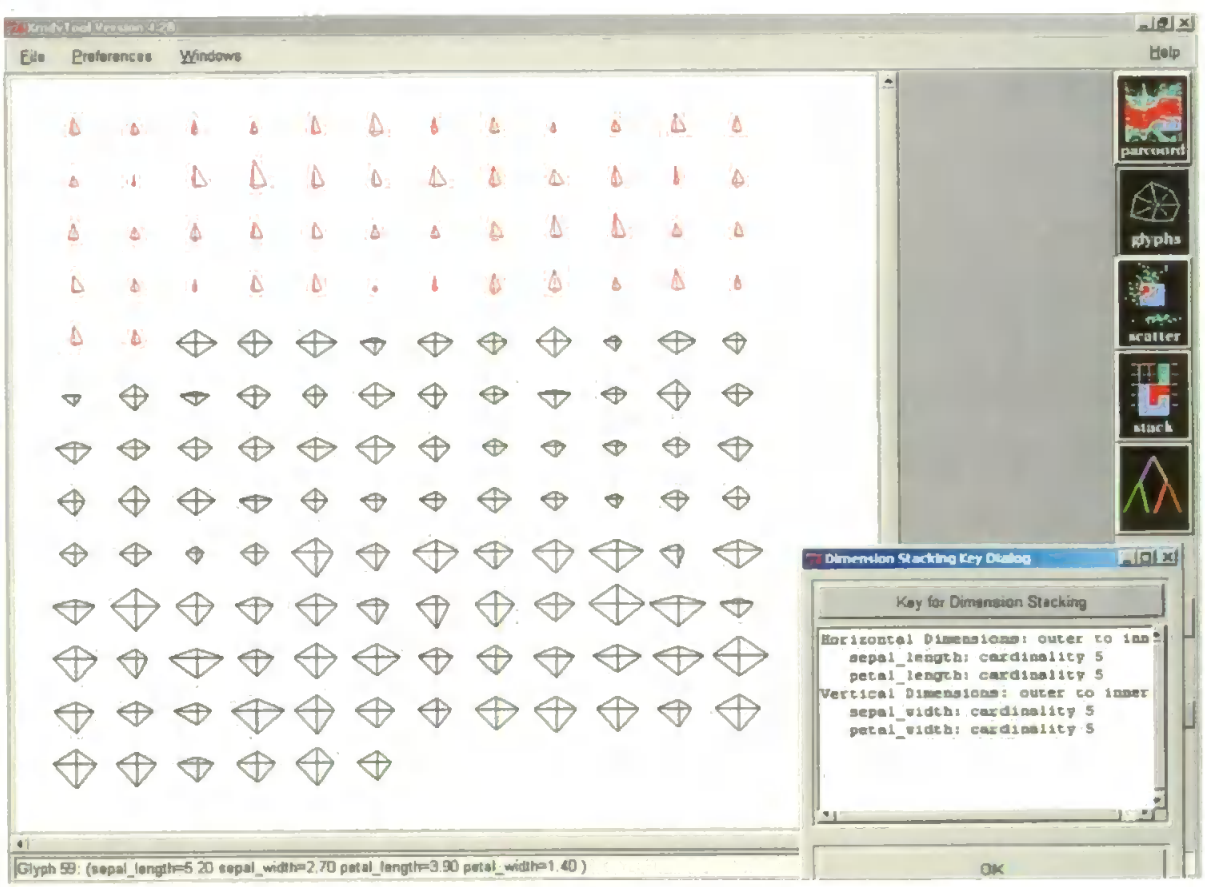


Figure 3.7: Star glyph display by XmdvTool (Ward 1994). Used with permission.

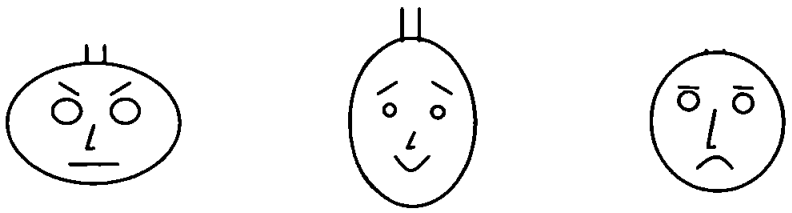


Figure 3.8: Examples of Chernoff Faces. Size and shape of face, eyes, nose, mouth, eyebrows & hair could all signify different variable values.

3.3.4 Embedded Axes

Another attempt to display multivariate data in a single display is to embed each variable within other variables. This means that an order for the variables needs to be chosen beforehand, hence the one-dimensional version of the embedded axes family is termed Hierarchical Axes (see Wong & Bergeron (1997, pp. 18-19)). Each variable is represented as a histogram with a user-defined number of bins. Histograms for the 'faster' variables are drawn inside the histograms of the 'slower' variables. Figure 3.9 shows the values of x (fastest) in dark grey, y in light grey and z (slowest) in white. The two-dimensional version is called Dimension Stacking (Leblanc *et al.* 1990). Again variables are embedded inside each other so the inner squares are a combination of all the variables and brushing is used to link the data. A similar technique called Autoglyph by Beddow (1990) (see Keller & Keller 1993, p. 178) takes advantage of a human's pattern recognition to see clusters in regions of the display. However these displays take some time to learn and it is necessary to change the order of the axes to see different attributes of the data.

A three dimensional version of embedded variables is the Worlds Within Worlds model (Figure 3.10). Feiner & Beshers (1990a) introduced n-Vision, an interactive display incorporating sets of axes within other axes. The values of the outer variables need to be fixed, and then the objective function (or dependent variable) can be calculated from the other two free variables and shown as a surface plot. The authors have incorporated a

number of interactive tools, but the main problem is that users need to know what they are looking for, as most of information is not visible in the initial display. The updated version called AutoVisual added a rule-based user interface, turning the system into a datamining tool as the criteria for search needs to be determined beforehand (Beshers & Feiner 1993).

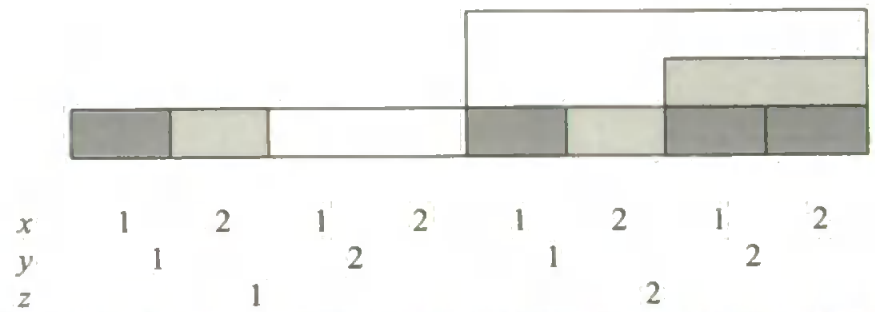


Figure 3.9: Hierarchical Axes display, the variable *x* is embedded inside *y*, they are both inside *z*. The points (1,1,1), (1,1,2), (1,2,2) and (2,2,2) are shown indicated by the dark grey boxes.

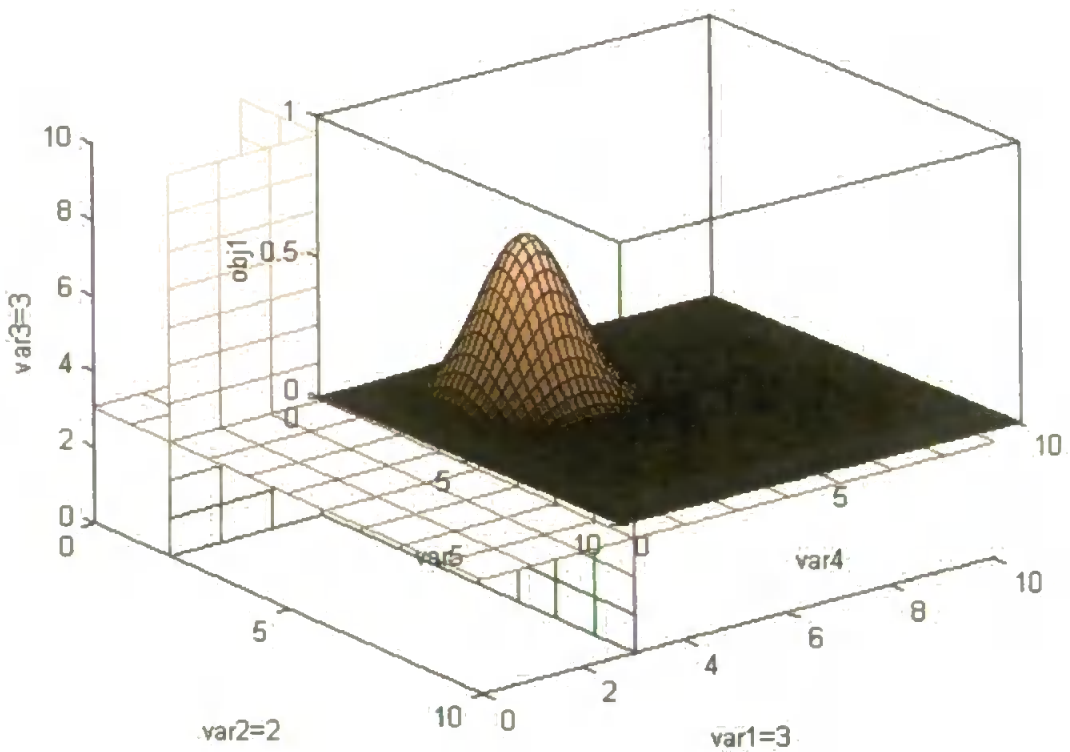


Figure 3.10: Worlds Within Worlds. Variables 1, 2 and 3 are fixed, 4 and 5 are changeable to give 'obj1' values, shown as a surface plot.

3.3.5 Parallel Coordinates

The problem with many multivariate visualisation techniques is that trying to capture all the information in one picture or set of pictures requires imagination or training by the human. Parallel coordinates is a visualisation technique that shows all the data in one picture and each dimension is represented equally. Figure 3.11 shows the Iris Data displayed using parallel coordinates by XmdvTool. The variables are arranged vertically in parallel while lines joining the relevant axes (or coordinates) represent an individual data item. A straight line in Cartesian space corresponds to the intersection of lines in parallel coordinates. Inselberg & Dimsdale (1994a) describe a number of geometric properties of parallel coordinates and applies them to collision avoidance in air traffic control (Inselberg & Dimsdale 1994b). As shown in Figure 3.11, brushing can also enhance the information presented in parallel coordinates. Drawbacks of parallel coordinates are that with a large number of data points and variables the lines overlap and darken making the plot difficult to read. Occlusion again becomes a problem with a large number of data points, the user will have to choose which variables and data points to view on top. Swayne *et al.* (1998) choose a subset of variables to display in parallel coordinates. The problem of darkening can be partially solved by colouring subsets of data and replacing clusters of lines by the average or centroid of the cluster (Siirtola 2000).

3.3.6 Animation

Wong and Bergeron (1997) also mention animation as another visualisation tool. Often these are interactive and animated versions of the multivariate tools mentioned above. The Grand Tour method by Buja & Asimov (1986) used time as one of the dimensions, so the display would change as the time parameter changes.

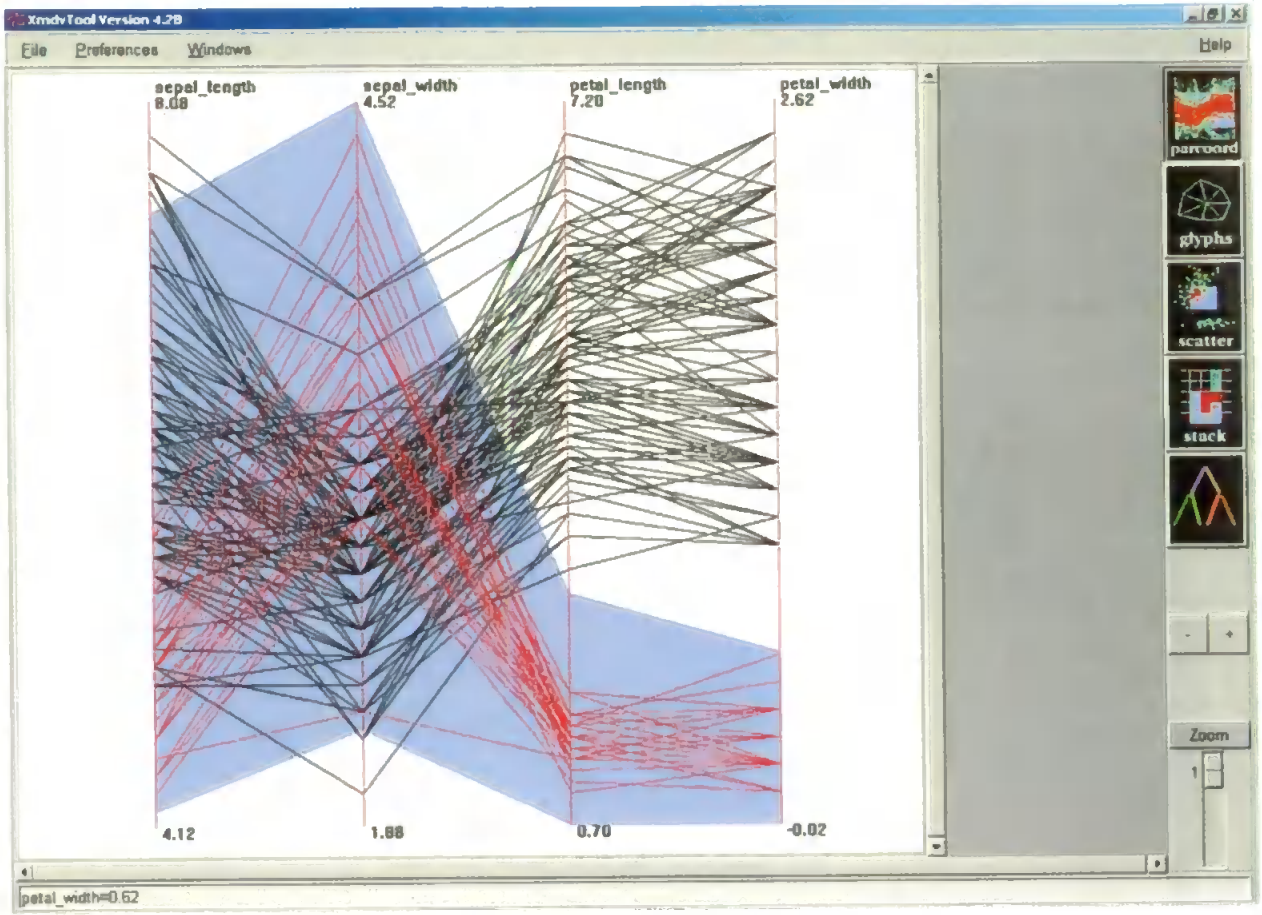


Figure 3.11: Parallel Coordinates of the Iris Data by XmdvTool (Ward 1994), used with permission.

3.3.7 Summary

This overview reveals the diversity of classical visualisation techniques that have been developed due to the increasing power of modern technology. Most visualisation systems incorporate a number of these tools as separate or combined displays. For the visualisation of engineering design data using evolutionary computation, it is suggested that the user does not wish to spend time understanding abstract symbols but will want to interact directly with data that is presented in an intuitive manner. The most intuitive displays that allow easy interaction are two-dimensional scatterplot matrices (Chambers *et al.* 1983) and parallel coordinates (Inselberg & Dimsdale 1994a, 1994b). Brushing (Becker & Cleveland 1987) is a useful enhancement to both these displays, as demonstrated by Hyperslice (van

Wijk & van Liere 1993) and interactive parallel coordinates (Siirtola 2000). Distortion controlled by the user can also enhance scatter displays in two or three dimensions (Carpendale *et al.* 1997), although the exact position of data in three dimensions may be unclear to the user. Most state of the art interactive displays use scatterplot matrices and parallel coordinates with brushing to depict multivariate data, although some authors recommend limiting the number of linked displays to three and filling the whole screen with those displays (Jern *et al.* 2003, Brodbeck & Girardin 2003). The rest of this chapter will review techniques to help visualisation for engineering design applications including sophisticated statistical techniques such as principal component analysis (PCA), independent component analysis (ICA) and clustering to enhance the information presented to the user.

3.4 Multivariate Statistical Analysis

3.4.1 Introduction

Multivariate data may exhibit complex structure that cannot be inferred by looking at or manipulating the original data. Statistical analysis is a form of exploratory data analysis that tries to match the structure of the data to known statistical models. Clustering is another type of data analysis technique that typically describes the arrangement of data in terms of clusters using a certain metric, usually Euclidean distance. Statistical and clustering techniques are closely linked but because of the large number of examples they are presented here in two separate sections. The techniques are demonstrated on the Iris Data described in Section 3.3.1.

3.4.2 Statistical Tests

Statistical tests on a set of data measure the extent individual items vary from some pre-defined acceptable measure; if not acceptable the data (or process) is termed 'out of control'. Usually the data is assumed to have a normal distribution and any data not within 95% of the centre of that distribution is out of control. For a single variable the *t*-distribution can be used (using n-1 degrees of freedom where n is the number of data points).

Two scatter plots show the Iris Data in Figure 3.12. The lines show the variance of the data projected onto each variable; they intersect at the mean value or centroid of the data. The outliers are obtained by a *t*-test on 95% confidence limits for each variable, that is those points of a variable x_i that lie outside the bounds $\bar{x}_i \pm \sigma_i * t$ where $\bar{x}_i$ is the mean, σ_i is the standard deviation and t=1.97 for 149 degrees of freedom (virtually infinite). These outliers are due to the sepal length and width variables only. However this test is not a good test for multiple variables especially if the variables are correlated (Jackson 1991).

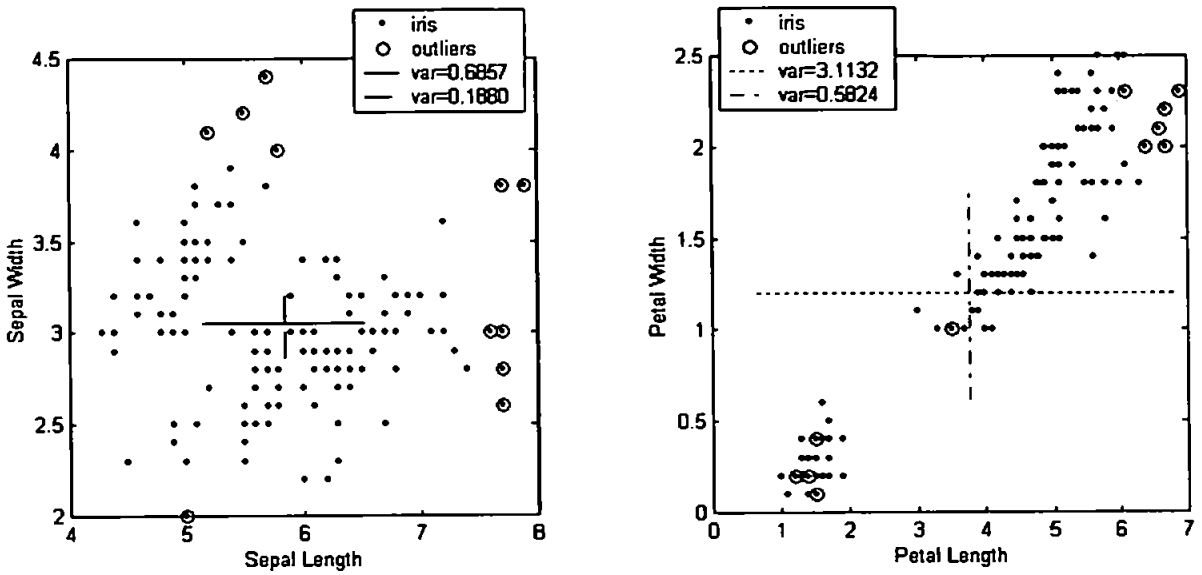


Figure 3.12: The Iris Data. Variances (var) of each variable are given and shown to scale on the plots, centred at the mean (centre of mass) of the data. Outliers outside 95% confidence limit according to the *t*-test computed for each variable are also shown.

3.4.3 Principal Component Analysis (PCA)

A multivariate *t*-test has been devised by Hotelling (1947) (see Gnanadesikan 1997, p. 82 and Jackson 1991, pp. 21-23) that uses principal component analysis (PCA) to define the outliers. The principal components are an orthogonal set of vectors that are the eigenvectors of the covariance matrix of a set of data. For a matrix of data **X** with v variables and n data points the covariance matrix **S** is defined by:

$$\mathbf{S} = \begin{bmatrix} s_1^2 & \cdots & s_{i1} & \cdots & s_{v1} \\ \vdots & \ddots & s_{ij} & \iddots & \vdots \\ s_{1j} & s_{ij} & s_i^2 & s_{ij} & s_{vj} \\ \vdots & \iddots & s_{ij} & \ddots & \vdots \\ s_{1v} & \cdots & s_{iv} & \cdots & s_v^2 \end{bmatrix} \qquad \textit{Equation 3.1}$$

Where s_i^2 the variance of variable x_i (i=1,...,v) and the individual covariance s_{ij} is given by:

$$s_{ij} = \frac{n\sum_k x_{ik}x_{jk} - \sum_k x_{ik}\sum_k x_{jk}}{n(n-1)} \qquad \textit{Equation 3.2}$$

The eigenvectors form the transformation matrix $\mathbf{U}$ such that $\mathbf{U'SU} = \mathbf{L}$ where $\mathbf{L}$ is a diagonal matrix. The diagonal elements of $\mathbf{L}$ are called the eigenvalues of $\mathbf{S}$. The determinant of the covariance matrix $|\mathbf{S}|$ is called the generalised variance; the square root is proportional to the volume generated by normalised data (Jackson 1991, p. 13).

The first principal component is the vector of largest variance through the data; the magnitude of the variance is equal to the corresponding eigenvalue. The second principal component is the vector with the second largest variance through the data with the constraint that it is orthogonal to the first, and so on. The projections of the data onto the principal components are called the *z*-scores: $\mathbf{z} = \mathbf{U'}(\mathbf{x} - \bar{\mathbf{x}})$. The *y*-scores are the normalised *z*-scores so that the mean ($\bar{\mathbf{y}}$) is zero and standard deviation σ is one. The *y*-scores are used to calculate the multivariate T^2 statistic: $T^2 = \mathbf{y.y'}$. The whole process is out of control if it exceeds a value found from the tables using the *F*-distribution:

$$T^2_{p,n,\alpha} = \frac{p(n-1)}{n-p} F_{p,n-p,\alpha} \qquad \textit{Equation 3.3}$$

Hence for the Iris Data with p=4, n=150, $F_{4,146,.05} \approx 2.425$ for 95% confidence, so $T^2_{4,150,.05} = 10.45$. Any value of $\mathbf{y.y'}$ that exceeds this value is regarded as an outlier. Figure 3.13 shows the *z*-scores and the corresponding eigenvalue (or variance) of each principal component as well as the outliers. Figure 3.14 shows the same outliers and the directions of the principal components on the original axes. In general the first few principal components can be used to describe most of the variance of the original data, the magnitude of each eigenvalue gives an idea of its importance. However if some of the principal components are used to reconstruct the original data, a residual analysis of the lesser principal components should be performed to check that no significant information has been discarded.

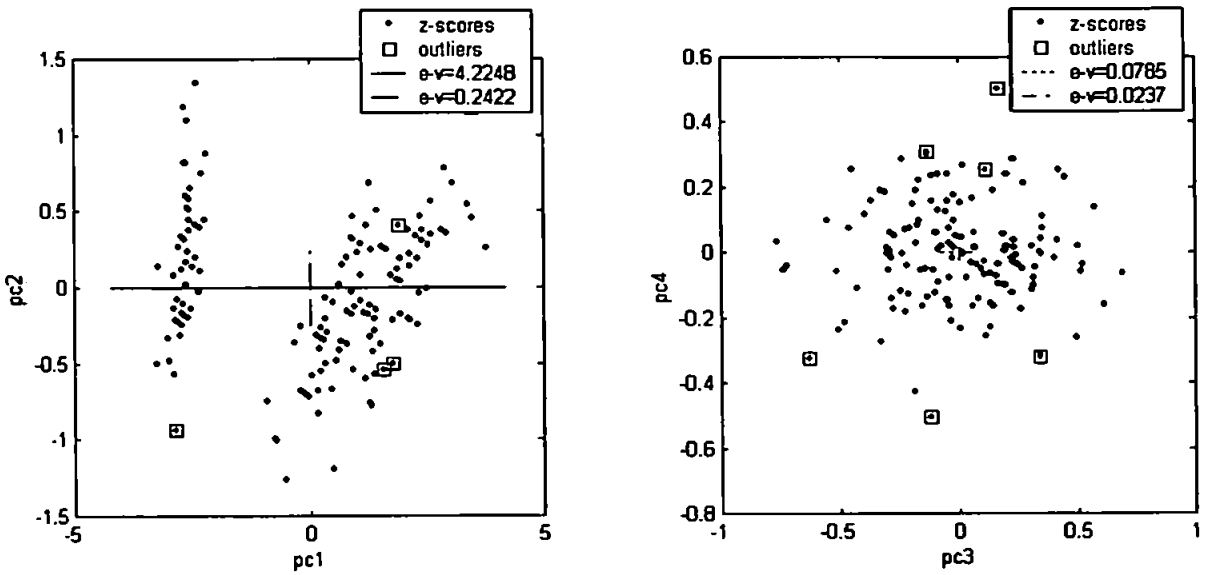


Figure 3.13: Principal components of the Iris Data showing the magnitude of eigenvalues (e-v) represented as a line. The outliers (highlighted by squares) are due to the T^2 statistic.

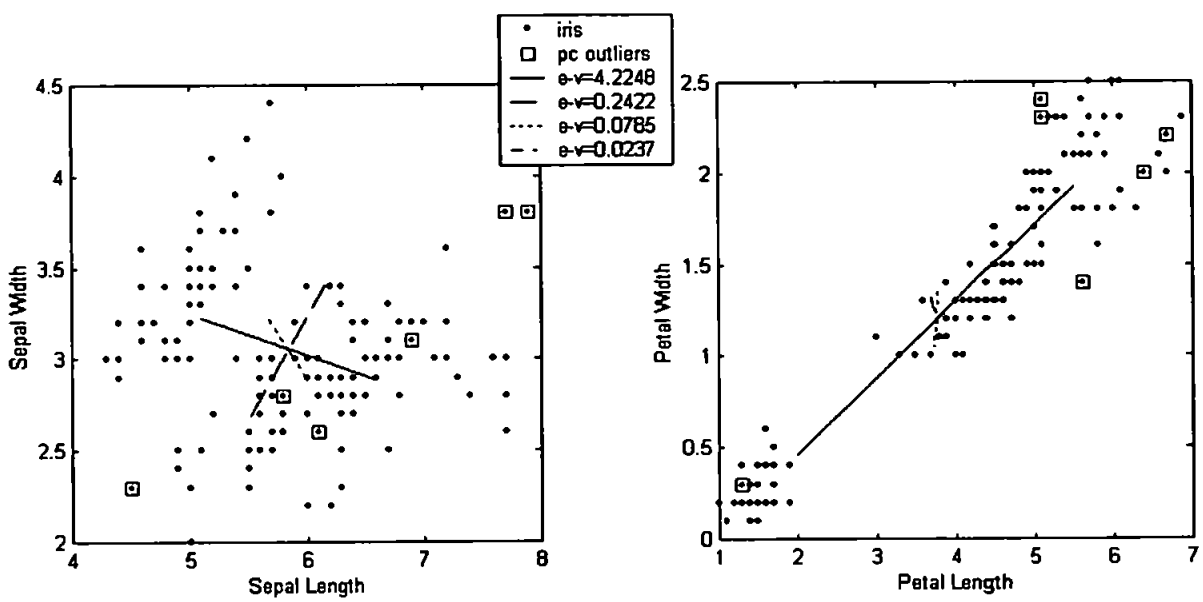


Figure 3.14: Principal component directions shown as lines on the original Iris Data axes, with eigenvalues (e-v) and outliers as in Figure 3.13.

3.4.4 Independent Component Analysis (ICA)

Independent component analysis (ICA) was developed by Jutten and Herault (1991) to tackle blind source separation (BSS). As described in Hyvärinen & Oja (2000) one of the applications of blind source separation is in solving the cocktail party problem that

separates the mixtures of signals in a noisy room so that the original signals can be detected. ICA was found to be successful at signal separation applications but it turned out that the technique used was very similar to projection pursuit (Friedman & Tukey 1974), despite the difference in application (see Section 3.4.5).

In general ICA uses an optimisation routine to find signals or vectors in the space so that the projection of the data onto the vector has a distribution that is maximally different from the Gaussian distribution (termed "nongaussianity"). There are various ways of measuring the nongaussianity of a projection. The classical definition of nongaussianity is Kurtosis; other approaches include using negentropy and maximum likelihood estimation (MLE). The FastICA algorithm (Hyvärinen 2003) uses a mixture of Kurtosis and negentropy to estimate nongaussianity.

A result of the FastICA algorithm on the Iris Data is shown in Figure 3.15. The outliers were found by performing a T^2 test as in the previous section. The values of k are the Kurtosis of the projection. The Kurtosis is the fourth central moment of a set of data (where the second central moment is the variance and the third is the skewness of the distribution). For normalised data u:

$$k(u) = E(u^4) - 3 = \frac{1}{n}\sum_{i=1}^{n} u_i^{\,4} - 3 \qquad \textit{Equation 3.4}$$

Gaussian data has a Kurtosis value of zero, a positive Kurtosis value indicates data that is thinner than Gaussian, with more data near the mean (super-Gaussian), while negative Kurtosis values indicate a flatter distribution than Gaussian (sub-Gaussian). As can be seen in Figure 3.15 the independent component with a negative Kurtosis value reveals the separation between the *Iris Setosa* data and the rest. Figure 3.16 shows the independent components and outliers on the original axes.

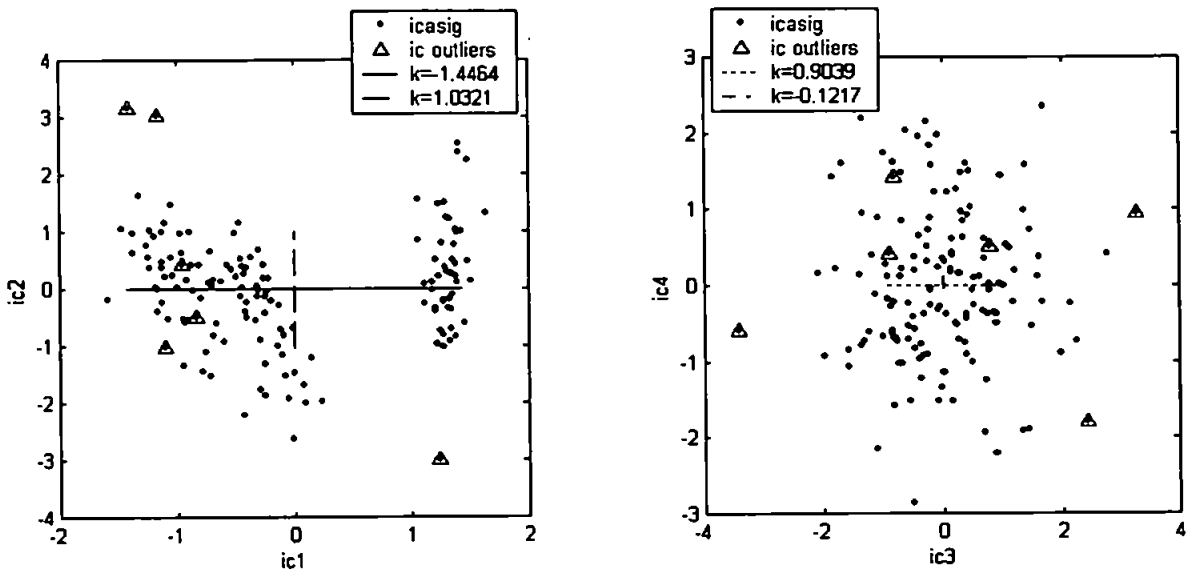


Figure 3.15: Independent components, outliers and kurtosis (k) values. Kurtosis is a measure of how closely the data resembles a Gaussian distribution. Computed using the FastICA algorithm (Hyvärinen 2003), supplied under the GNU (2003) licence.

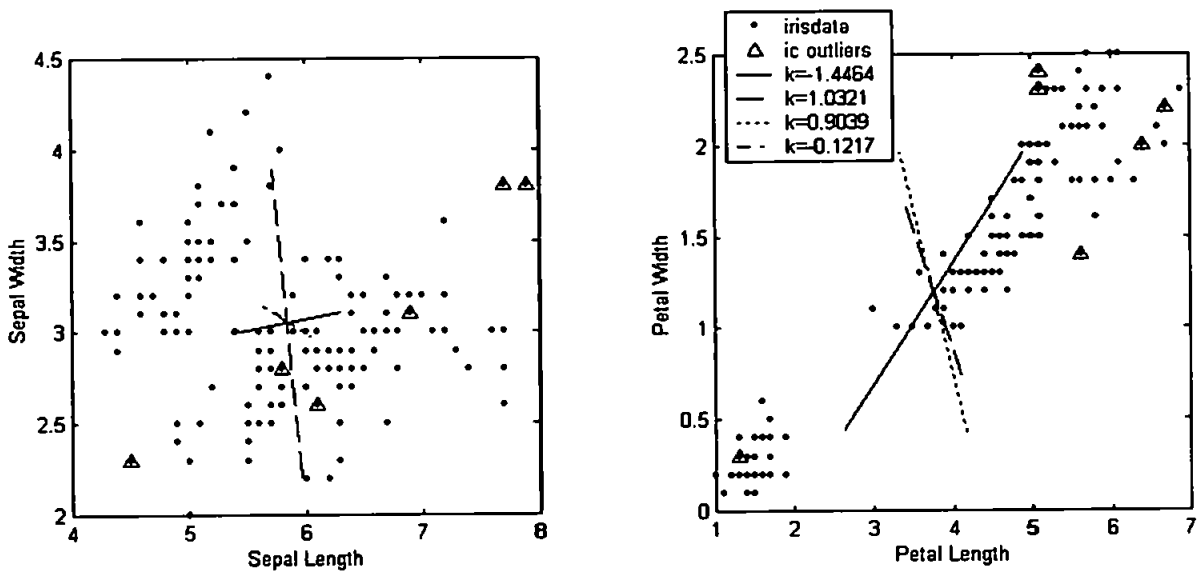


Figure 3.16: Direction of the independent components shown on the original axes, the value of Kurtosis, k, is shown.

In the case of the complete Iris Data there is little difference between the outcome of PCA and ICA. The *Iris Setosa* class is easily identified from the original variables. If the *Iris Setosa* cluster is removed the advantage of ICA over PCA can be seen. Figure 3.17 shows the result of PCA analysis, which does not reveal any separation of the data. One of

the ICA results (different results are obtained each time) shown in Figure 3.18 reveals a separation in the independent component with a large negative kurtosis value. The line ic1=-0.25 separates the *Iris Virginica* and *Iris Versicolor* clusters up to a few exceptions, which is the best that can be expected from a linear approach.

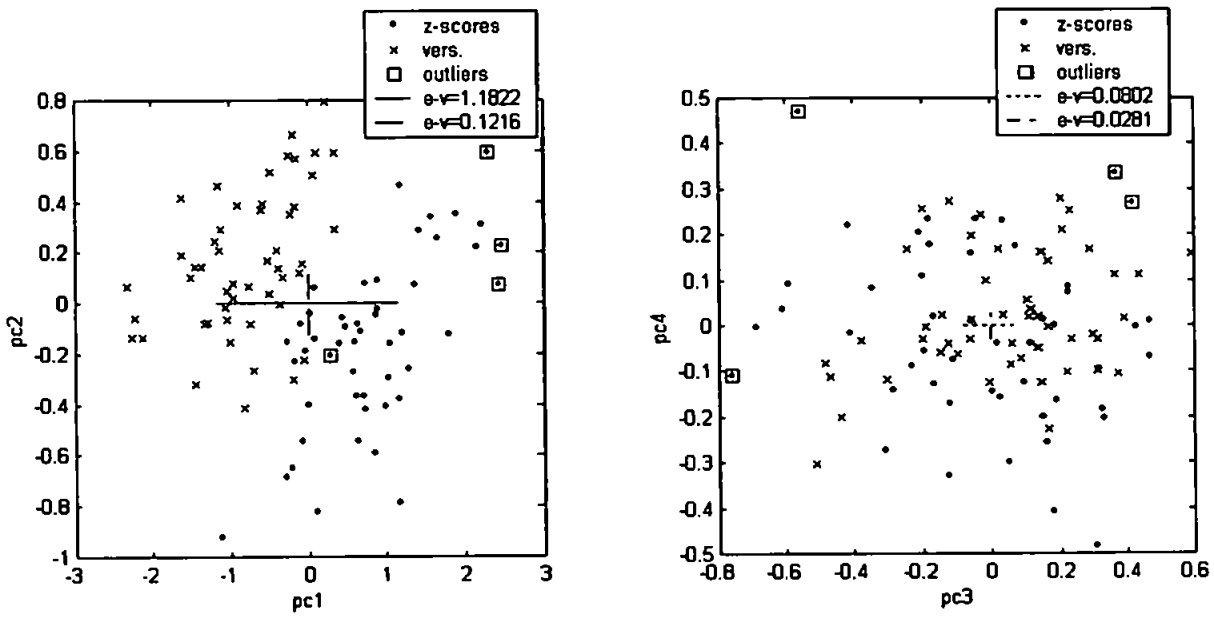


Figure 3.17: PCA on the Iris Data with *Setosa* removed. *z*-scores, outliers and eigenvalues (e-v) of this data are shown, the *Versicolor* cluster is partially separated from the rest of the data by the first principal component (pc1).

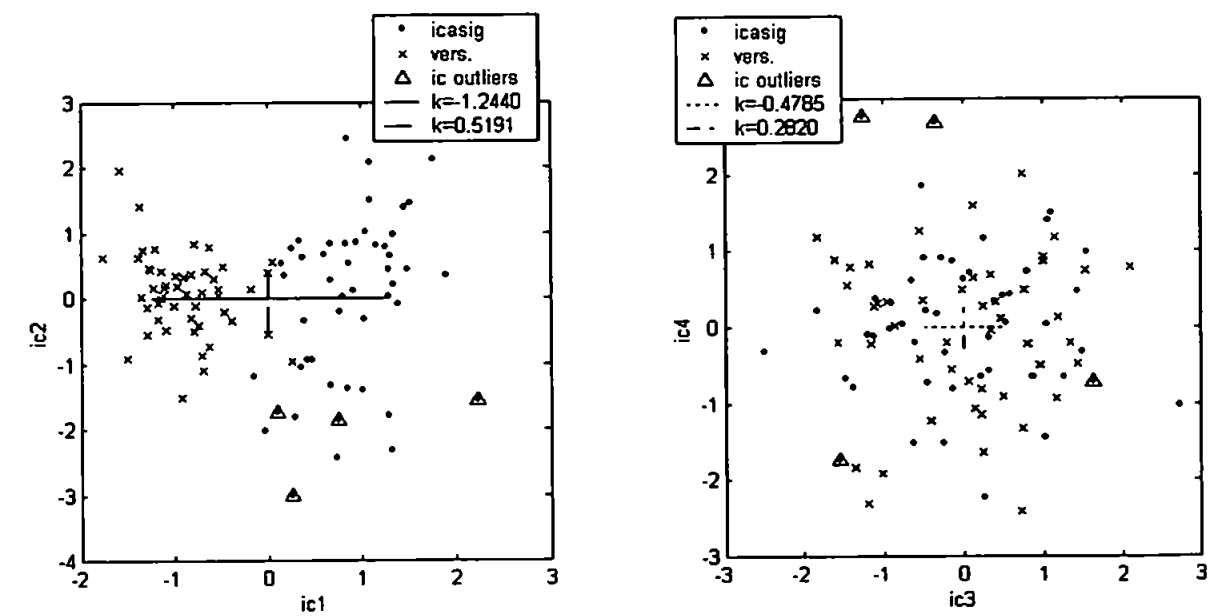


Figure 3.18: ICA on the Iris Data with *Setosa* removed, outliers and value of kurtosis, *k* are shown. ic1 visibly separates *Versicolor* from the rest of the data up to a few exceptions.

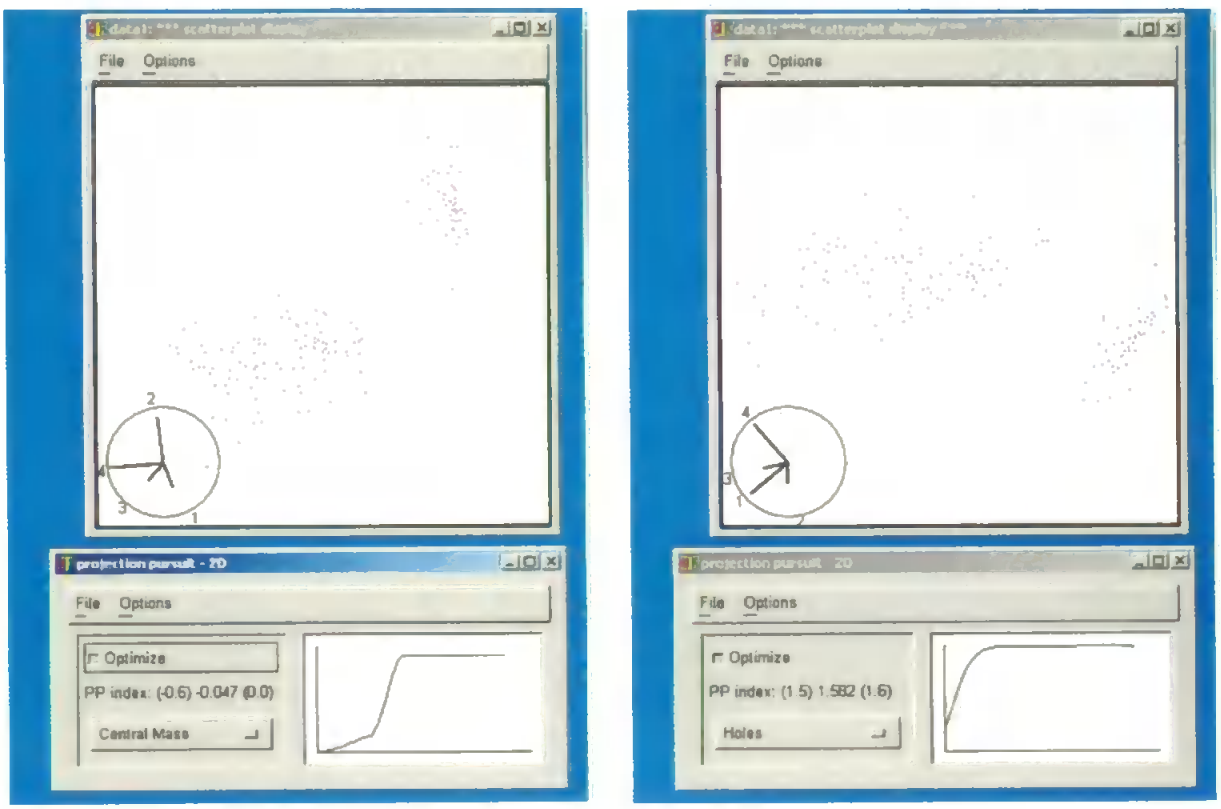
3.4.5 Projection Pursuit

PCA and ICA are both special cases of a procedure called projection pursuit by Friedman and Tukey (1974). Projection pursuit attempts to find 'interesting projections' in a set of data, these projections (unlike in PCA) do not have to be orthogonal. ICA as described in Hyvärinen & Oja (2000) is a special case of projection pursuit whose goal is to maximise the "nongaussianity" of the projections.

The original version of Projection Pursuit by Friedman & Tukey (1974) maximised two attributes of the data – the variance of the data and a local density measure. These criteria are motivated from their experience that the most interesting projections keep the overall spread of the data high while searching for a number of small local clusters. The outcome of the iterative algorithm depends on the choice of the initial vectors. The algorithm seeks distinct vectors, but they do not have to be orthogonal. The idea is that they are independent due to the requirements of the optimising procedure rather than the original variables, although in practice they are not completely independent (Hyvärinen *et al.* 2001).

Projection pursuit has been taken further by Cook *et al.* (1993, 1995) and incorporated into the visualisation tools XGobi (Swayne *et al.* 1998) and GGobi (Swayne *et al.* 2001). Cook *et al.* (1993) noticed that different projection pursuit indices could be used to find different attributes of the data. In addition the order of the index can also be used to find a varying amount of detail in the data. Rather than trying different projection pursuit indices on the data, one general index was devised and truncated so that the order of each index could be observed. Indices of zero order have optima at projections either at a central mass or a hole in the data. Indices of the first order have optima on projections of skewed data. As the order increases the optima become more difficult to find, although they reveal very fine detail in the data if they are found. During optimisation the data is

projected onto one or two dimensions because they are the most natural for humans to visualise. Close human supervision of projection pursuit techniques is advised, particularly as different optima will be revealed in each run (Cook & Buja 1997). Figure 3.19 shows two projections found by optimising different indices that look very similar.



(a) (b)

Figure 3.19: Projection Pursuit on the Iris Data using the GGobi interface (Swayne *et al.* 2001). The data is rotated searching for both Central Mass (a) and Holes (b) in the data, another option is to search for skewed data. Screen shots used with permission.

3.4.6 Other Optimisation Routines (MM, EM and MDS)

Similar to ICA, mixture models (MM) attempt to match the data directly to mixtures of Gaussian shapes. The number of Gaussians that match the data need to be estimated, then the parameters, mean and variance of the Gaussian are optimised to match the data. This method assumes the data can be modelled using Gaussian distributions. Expectation maximisation (EM) can be used to optimise the algorithm by computing the likelihood that the data can be modelled by a certain number of Gaussians.

Multidimensional scaling (MDS) is another optimisation routine used to depict high dimensional data in lower dimensions. SAMMON mapping by Sammon (1969) was one of the first techniques developed that attempts to map the Euclidean distance between all data points to a two-dimensional view. SAMMON mapping is closely linked to Self-Organising Maps (SOMs) that is a neural inspired method to cluster and visualise relationships between data or attributes of objects; indeed Kohonen (1997) suggested using SAMMON mapping as a preliminary step before using SOMs or to quickly check the result. Dimensionality reduction can be computationally expensive for a large number of data points, but the low-dimensional representation is very popular and has lead to a number of applications (König 2001, Buja *et al.* 2001).

3.4.7 Summary

The fastest methods to get some understanding of high dimensional data are statistical tests and their derivatives such as principal component analysis (PCA). Usually the first two principal components are displayed so the projection onto the vectors of largest variance can be inspected. Independent component analysis (ICA) is a special case of projection pursuit that has been shown to be a successful approach to finding clusters in the Iris Data set. It is worth considering other approaches to projection pursuit that optimise specific characteristics of a data set a user is interested in, particularly data containing a hole, central mass or skewed data. The main problem with using projections from PCA or ICA is that the data has been transformed to abstract variables that show interesting attributes of the data but the relationship with the original variables may be unclear. High-dimensional data can even be represented on a two-dimensional picture by optimising the mapping of the distance between points (for example multidimensional scaling (Buja *et al.* 2001)) but this does not help the user understand which variables form the clusters. Expectation maximisation and Gaussian mixture models can also be used to guess the shape of the data.

3.5 Clustering

3.5.1 Introduction

Clustering is the art of grouping data into subsets due to some predefined criteria, usually without any preconceptions about the data. Classifying usually means the process has been supervised (Gnanadesikan 1997), so that data is put into a group that has already been labelled. A general clustering algorithm cannot do labelling, as it requires input by a user or domain knowledge; this input may be contrary to the 'natural' clustering of the algorithm.

To start using a clustering algorithm, a pattern representation and the distance metric between patterns need to be chosen. The best pattern representation is one that closely resembles the data to be classified, for example an attribute such as colour, shape or position in space. If the engineering design data is given in the form of continuous Cartesian data then the distance matrix can be constructed from the Euclidean distance between the points. However this may not be the best representation to choose, for example curvilinear clusters may be best represented in polar coordinates (Jain et al. 1999). The shared nearest neighbour technique (see Section 3.5.4) uses the number of nearest neighbours as a distance metric and this works very well. However engineering design data is often of a discrete or discontinuous nature, so a different definition of distance may be needed. Additionally the objective(s) may have more importance than other attributes of the data so further information may be required such as using the distance from the fittest point or giving a larger weight to the objective attribute(s).

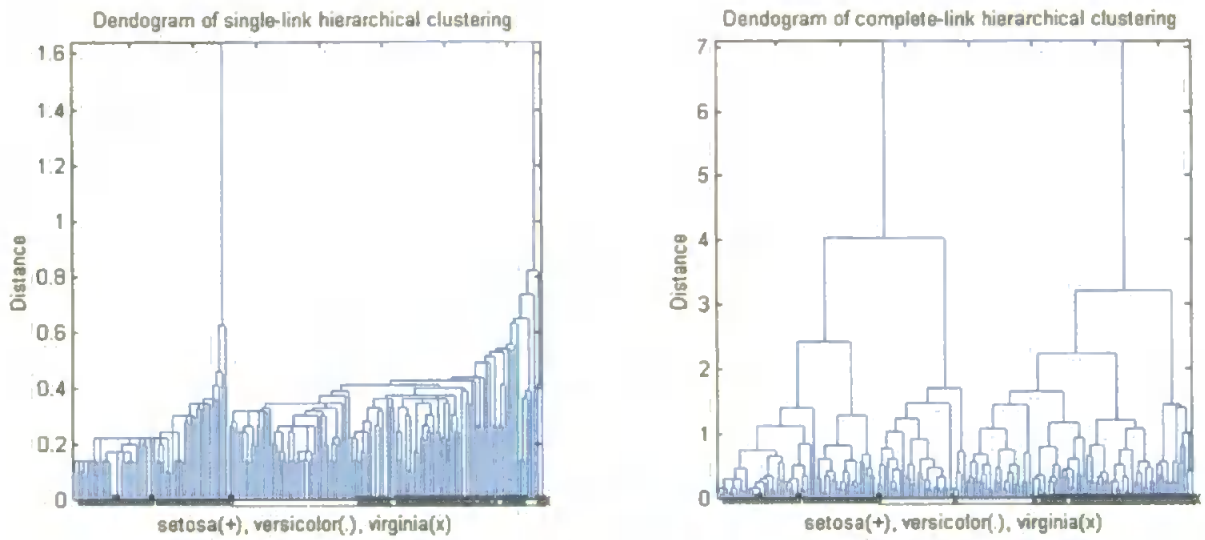
The main part of the clustering algorithm is the *grouping process*; there are two main approaches: partitional and hierarchical methods. Partitional algorithms assume a certain number of clusters and attempts to find an optimal partition that agrees with the number of clusters, on the other hand hierarchical clustering generates links between data

points according to some criterion and leaves the grouping process to the user, usually by way of some visualisation technique.

3.5.2 Hierarchical Clustering

The philosophy behind hierarchical algorithms is that data points are not assigned to clusters during the process, but after clustering the number of clusters can be chosen by the user by splitting the output at a certain 'level'. Thus a cluster is composed of smaller clusters and can be continually decomposed until each cluster is singleton. Often a "dendogram" is used to view the hierarchical output in a tree-like manner, sometimes enabling interaction with the user. An example of a dendogram after clustering on the Iris Data is shown in Figure 3.20 (see Jang (2003) for MATLAB code to generate the cluster and the dendogram). Usually the clustering is done agglomeratively such that all the points are initially in separate clusters and the two 'closest' sets are merged, continuing until all the data points are assigned to one cluster. The opposite, divisive, method is to start with all the data in one cluster and continually break the cluster at the point of largest separation until the number of clusters is the same as the total number of items.

The criterion for merging two points during agglomerative clustering can affect the look of a dendogram in quite a big way. At any stage in the clustering, the 'distance' between one cluster to another is calculated and the two with the minimum distance are merged. This distance could be the minimum distance between individual points in each cluster (single-link), the largest distance between individual points (complete-link) or other criteria (Jain *et al.* 1999). Figure 3.20a is a dendogram generated using the single-link algorithm while Figure 3.20b shows output from the complete-link algorithm on the same data. Both dendograms split the *Iris Setosa* data from the rest of the data, but the clustering looks very different. The single-link algorithm will return chain-like clusters, while the complete-link algorithm finds more compact and spherical clusters.



(a) (b)

Figure 3.20: Dendograms of clustered data. The actual data labels are shown at the bottom. Single-link (a) and complete-link (b) hierarchical clustering is shown. See Jang (2003) for MATLAB code to generate the dendogram, used with permission.

This representation may be useful in engineering design applications as all the points can be presented at different levels of clustering and the user could choose which level to look at. Hierarchical clustering using a dendogram gives a good visual representation of the data, although for large data sets the computation time is prohibitive (Jain *et al.* 1999) and the picture becomes very heavy with lines (Gilbert *et al.* 2000), mirroring the problems with parallel coordinates.

3.5.3 Partitional Clustering

The most popular clustering algorithm is the *k*-means algorithm, originally devised by MacQueen (1967), because of its efficiency, easy implementation and understandability. It is a partitional algorithm so that partitions are made and tested during the algorithm. Initially the number of clusters k of a data set X is estimated either by prior knowledge or randomly. The k centroids of each cluster $c_i \in X$ (i=1,…,k) are chosen at random and the

rest of the data is assigned to the cluster represented by its nearest centroid. The next set of centroids is then chosen by finding the point in each cluster that is nearest the centre of that cluster. In Cartesian data the centroid (mean value in each variable of the data) within each cluster could be used, but if only the distance metric is available then one of the actual points has to be chosen. One of many options is to use the minimax criterion; that is for all elements of a cluster, make a list of the maximum distance to the other points and choose the minimum value of that list. This will be the point nearest the centre of the cluster. The algorithm is repeated until the same cluster centres are found in subsequent iterations or a maximum number of iterations is reached. It is possible that the algorithm will oscillate between two cluster configurations and not converge, especially if the initial number of clusters does not match the data (Levinson *et al.* 1979). Figure 3.21 shows the result of *k*-means clustering on the Iris Data assuming the number of clusters is three. The number of misclassified points is relatively low (around 9), however this method was told there were three clusters at the start; the algorithm would require some pre-processing or repeated iteration with different parameter settings to compute the number of clusters automatically.

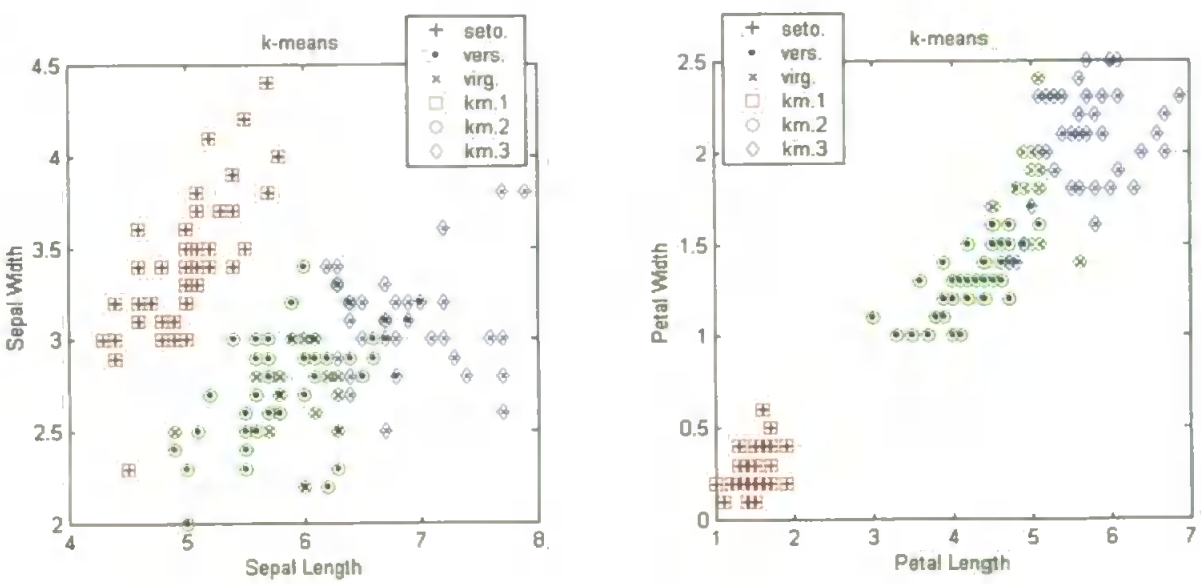


Figure 3.21: Clustering using the *k*-means algorithm, with *k*=3. The *Setosa* class is correctly classified, but there are misclassifications in the other two classes.

Levinson *et al.* (1979) used the chainmap algorithm in their interactive system to estimate the initial number of clusters. The chainmap algorithm simply picks an arbitrary point, finds its nearest neighbour and lists the other points according to the nearest neighbour to the previous one. Each member k of the chain is associated with the distance between k and k-1: d_k. The initial output of the chainmap is the plot d_k against k. Spikes in the plot reveal cluster boundaries. Well-separated boundaries usually show up in the plot, the algorithm is sensitive to starting point but is computationally inexpensive so multiple runs are possible. Other versions of partitional algorithms are density-based clustering and similarity graphs that assign dense or similar regions to the same cluster if they are connected to each other (Fasulo 1999).

3.5.4 Shared Nearest Neighbour Clustering

Clustering algorithms return the type of clusters they are asked to look for. Like many algorithms, the k-means will find spherical clusters if the Euclidean distance metric is used. Conversely the single-link hierarchical algorithm will find chain-like clusters. The shared nearest neighbour algorithm by Jarvis & Patrick (1973) clusters a data point due to its proximity or similarity to other points in the data. So points that are very close will be clustered together, but a number of sparse points may also be clustered if they have similar attributes. The algorithm uses two parameters. The first parameter is k, the number of nearest neighbours, that are listed for each point in the data set. Two points are assigned to the same cluster if the number of shared nearest neighbours in k exceeds a threshold k_t, the second parameter. A visual output of the clustering results is useful because a kind of hierarchy of clusters can be discovered using different input parameters. As k increases the number of clusters found decreases because the size of each neighbourhood is growing, however increasing k_t causes the clusters to become tighter and smaller because the number of shared nearest neighbours required to put the points into the same cluster is higher. Levinson *et al.* (1979) modified the technique so that overlapping clusters could be

identified. This algorithm can represent any type of cluster because the clustering criterion is based on similarity with other points rather than the actual distance metric used. Therefore it is more versatile than *k*-means, but the versatility can only be exploited by changing parameters and viewing the results. Figure 3.22 shows the result of shared nearest neighbour clustering with k=10 and k_t=4, again about 9 points are misclassified - although different to those misclassified by *k*-means. It is difficult to find a set of parameters that returns three clusters for this data.

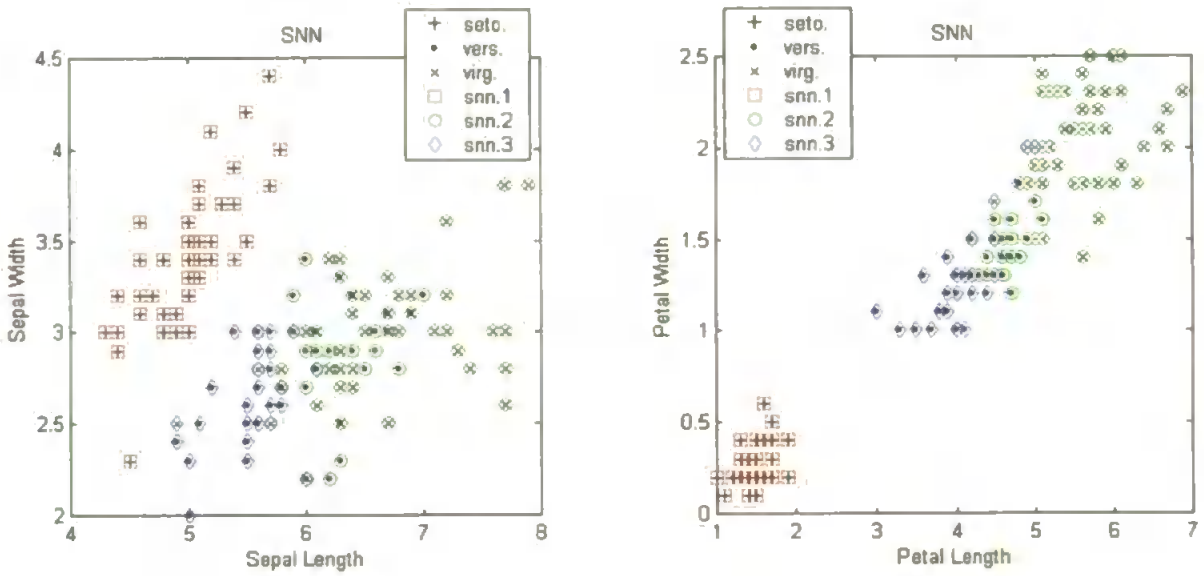


Figure 3.22: Clustering using shared nearest neighbour technique, k=10, k_t=4.

3.5.5 Hybrid Clustering

The *k*-means algorithm is better at dealing with large data sets because of efficiency but the number of clusters needs to be estimated beforehand, also outliers can cause *k*-means problems if they are not identified and removed or added to the correct cluster. Levinson *et al.* (1979) suggested using the chainmap and modified shared nearest neighbour to estimate the number of clusters before *k*-means was used in the ISODATA environment. ISODATA is an interactive clustering process where the human gets involved in identifying outliers and refining the clusters. Clusters are merged or split according to parameters input by the

user, there are a number of parameters to change but the increased flexibility results in very good clustering according to Levinson *et al.* (1979). Some of the properties of clustering identified by Levinson *et al.* (1979) for speech recognition are applicable to engineering design, namely a method that detects overlap among clusters, identifies outliers and allows interaction. To measure the amount of overlap between clusters they used the ratio of average intercluster to average intracluster distance.

Cluster presentation is important especially when considering data in high dimensions. Hinneburg *et al.* (1999) used a multidimensional grid (OptiGrid) to cluster data. Then visual techniques are used to find and display separators in data. Clusters are presented using colour and shape to represent their size and amount of separation from other clusters. The cluster separators are found by looking at the projections of the data onto one or two dimensions. A technique by Agrawal *et al.* (1998) described in Fasulo (1999) also breaks the data down into subspaces, the algorithm complexity is thus exponential with the number of dimensions. These techniques can only serve as a starting point for finding true clusters in the data as the user will need to combine the knowledge from each subspace to discover whether the data breaks down further. Ng & Han (1994) introduced CLARANS to cluster very large data sets, mainly by sampling the space and clustering the samples.

3.5.6 Summary

Clustering techniques give a more refined view of the data at some computational expense. The main computational expense is in computing the distance matrix between all points in the data to be clustered. For a large number of points calculating all the distances becomes prohibitive. It may be necessary to choose a subset of solutions found during a GA run to efficiently cluster the data. Once the distance matrix has been found the clustering techniques are usually relatively fast. The choice of a particular algorithm depends on the

application. Hierarchical algorithms have no parameters and can be quickly viewed in the form of a dendogram for further user interaction. However for a large number of data points dendograms have been found to be less useful. Partitional clustering algorithms and shared nearest neighbour technique need parameters such as number of clusters to work efficiently. Some of these parameters could be found off-line by repeated iteration of the algorithms and comparison with some error from the original data (Sierra & Corbacho (2000) did this with the *k*-means algorithm).

The brief experiments described in this section shows that the algorithms have only partial success in separating non-linear clusters. It can be concluded that any clustering algorithm may require parameter changes or user input to successfully partition arbitrary data, therefore speed and adaptability to engineering design data and the wishes of the user would be the main requirements of a clustering algorithm in an interactive system.

3.6 Relevant Work (State of the art)

This section investigates state of the art systems that have been built for or are applicable to visualisation and interaction with engineering design data. The main questions are what do these tools have and what do they lack, in particular with regard to visualisation of data provided by evolutionary computation.

Long experience in human computer interaction for engineering design led to the development of "Attribute Explorer" (Spence & Tweedie 1998) that allows change in output attributes as well as input of a problem. Colour is used to show interaction between attributes in the spirit of a brushing tool. A slider can be used to choose the required attribute in one graph and then elements in other graphs that satisfy the attribute are highlighted. The other elements are coloured depending on how many attributes they satisfy, see Figure 3.23. Attribute Explorer was designed to help users choose between

products to buy. The "Influence Explorer" (Tweedie *et al.* 1996a) is for making a product and is even more related to engineering design and the choices an engineer will have to make. Figure 3.24 shows the display of matching data concerning parameters and performance values grouped separately. This technique mixes parallel coordinates, histograms, brushing with colours and allows multiobjective satisfaction in a similar way to the Attribute Explorer. Again colour is used to identify which solutions satisfy the performance criteria and those that fail. Tolerances can be identified by changing the input parameters until the solutions change colour.

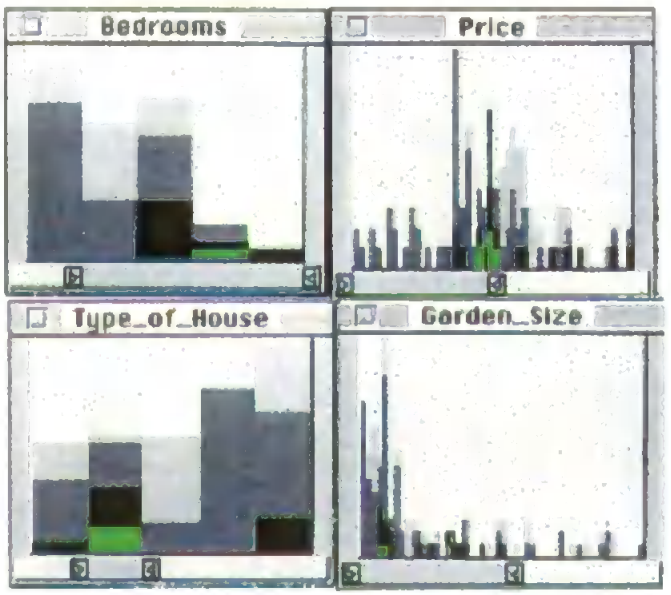


Figure 3.23: Attribute Explorer display, colour indicates the extent to which solutions satisfy the requirements. Reproduced from Spence (2003) with permission.

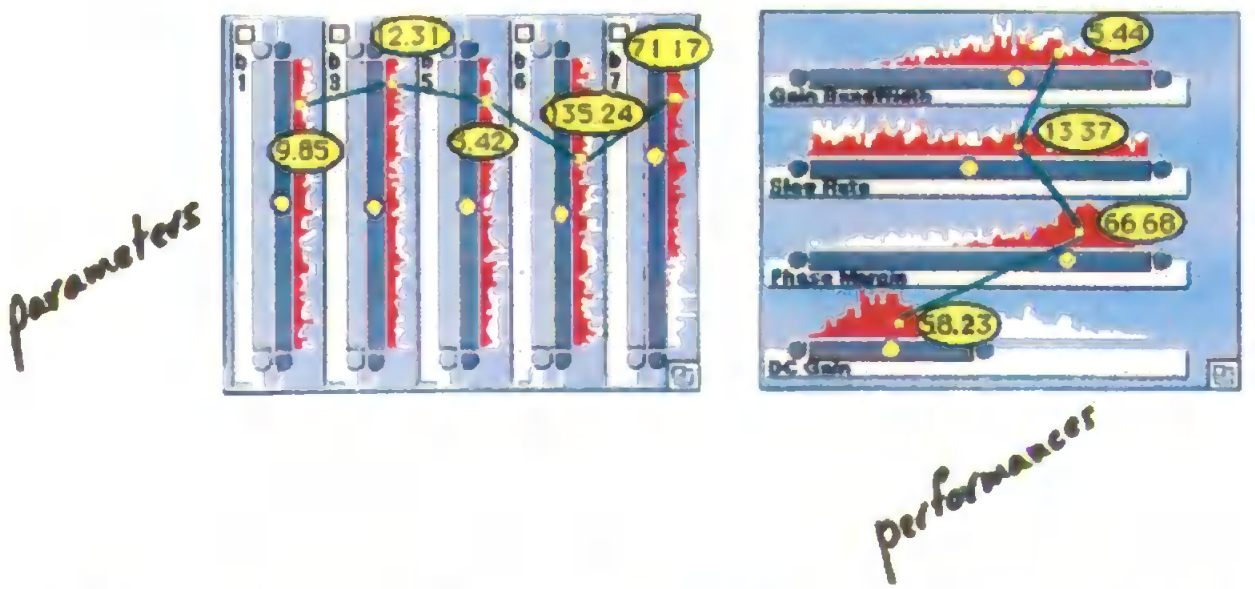


Figure 3.24: Influence Explorer display, Input parameters and output performances shown, again colour indicates satisfaction of performances, tolerance on each performance can be seen. Reproduced from Spence (2003) with permission.

A further interactive scheme by Spence is the "Prosection Matrix" described along with the other tools in his webpage (Spence 2003) and book (Spence 2001). The Prosection Matrix is a scatterplot matrix of 'parameters' (or variables) with an interactive box that enables the user to specify tolerances in each parameter. Figure 3.25 shows the scatterplot with designs coloured according to the number of 'performances' (or objectives) they satisfy. The solutions that satisfy all customer requirements are coloured green, while the overlaying yellow box is the tolerance allowed in each variable. Larger tolerances usually result in decreasing manufacturing cost (Phadke 1989, pp. 33-34), so the task for the user is to change the position and size of the yellow box so that the green area is covered as much as possible without catching too many infeasible solutions (Spence 2003). The trade off between reducing manufacturing cost and increasing the yield of feasible solutions can then be optimised. This tool again exploits a number of multidimensional visualisation techniques: scatterplot matrices, interactive brushing and colour coding in a similar way to Hyperslice (Section 3.3.2).

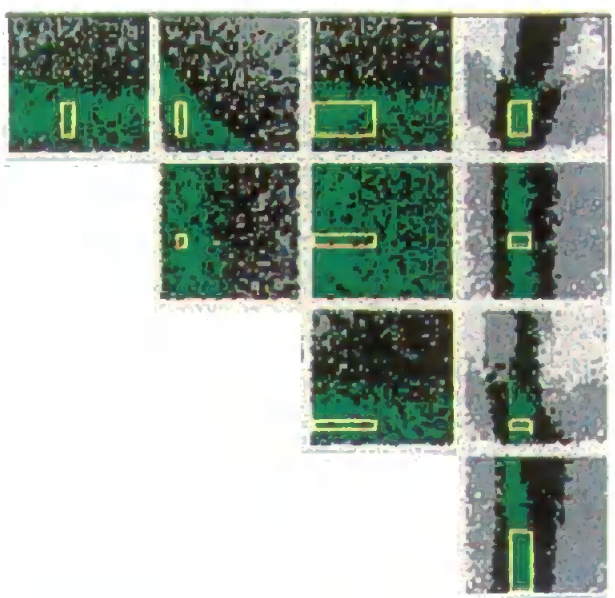


Figure 3.25: Prosection Matrix of parameter space (variables). Solutions that satisfy performances (objectives) are shown in green, the manufacturing tolerances are represented by yellow boxes. Reproduced from Spence (2003) with permission.

The Influence Explorer and Prosection Matrix were combined in an effective visualisation and engineering design tool with the result that "...these tools transform a very difficult cognitive problem into a much easier perceptual task." (Tweedie *et al* 1996b, CONCLUSIONS). A limitation of these systems is that solutions are generated randomly in specified regions which may result in missing information; without proper investigation an incorrect assumption of robustness could be formed, as mentioned in Section 2.2.4.

Andreas Buja and the teams at AT&T Labs have introduced a number of tools such as XGobi (Swayne *et al.* 1998), XGvis (Buja *et al.* 2001) and GGobi (Swayne *et al.* 2001). XGobi is the predecessor to GGobi, both of which contain a number of visualisation and interaction possibilities. Scatterplot matrices, parallel coordinates, brushing and changing colours are all supported. More sophisticated tools include the ability to compute and view the principal components and perform projection pursuit together with a grand tour as described in Section 3.4.5. During projection pursuit each variable is shown in the form of a 'widget' so that how much a certain variable (or principal component) contributes to the current projection can be evaluated. GGobi was updated with a familiar windows system that can be run on PCs. A number of windows displaying the same data in different ways can be opened and points between windows linked using brushing. Jittering, smoothing and many other features are present, subsets of variables can be viewed in parallel coordinate mode. The designers of GGobi initially over-designed the interface while they built up an understanding of various tools and data, it was then simplified as experience in the techniques was gained.

Gilbert *et al.* (2000) introduced Space Explorer that combines the use of a dendogram with PCA or multidimensional scaling. Various problems with clustering were identified such as poor scalability, possible misclassification and the difficulty with understanding dendograms of large data sets. Hence the PCA output was used alongside

the dendogram to identify clusters and view them in each display. A VRML display (virtual reality markup language) was also used to allow three-dimensional interaction around the data. Again this display suffers from not linking the abstract clustered view to the original variables. Subsequently Signature Explorer was formed (Noy & Schroeder 2001). The idea is to let the user interact with the data and tell the algorithm how they would cluster a subset of the data. The algorithm then matches the user's clustering preference to well-known clustering techniques and parameters. The project is in initial stages but the combination of visualisation and feedback seems a powerful way to cluster data using the human perception of similarity.

Harri Siirtola has extended two well-known techniques by adding interaction to parallel coordinates (Siirtola 2000) and allowing reordering of a matrix of data (Siirtola 1999). The parallel coordinate system implemented brushing but limited the update rate to 0.1 second so that an almost continuous feedback reveals aspects of the data that an immediate response does not. Alternative colours were used to highlight different clusters but a large number of polylines can occlude a parallel coordinate display, so interaction by zooming in on each variable was also implemented to reduce the amount of information. Furthermore instead of implementing hierarchical clustering which is computationally intensive, quick clustering techniques based on the user's input were used to replace a number of lines with the average line and 1 standard deviation bars shown on each dimension. Two subsets can also be compared by examining the polylines joining two variables. Different coloured bars on the display can represent positive and negative correlation. The "Reorderable Matrix" example allows the user to move variables whilst comparing individual solutions. This technique was used on tabular data, which is not relevant to engineering design, but allowing the user to change the order of a scatterplot matrix may help to understand the relationship between variables. Siirtola (2003) reports on a successful experiment combining parallel coordinates and the Reorderable Matrix

showing that coordinated views using different representations can enhance understanding of a problem.

3.7 Conclusions

Most systems designed to aid the visualisation of data and interactive engineering design are based on the traditional techniques of scatterplot matrices, parallel coordinates and linked brushing tools. Sometimes simple clustering techniques are also incorporated, with hierarchical clustering providing a useful visualisation alternative, although the dendogram, like parallel coordinates, can suffer from too many data points. PCA is often used to give a dimensionally reduced version of the data, resulting in nicely defined clusters but the relation between the clusters and the original variables is unclear. Other multidimensional scaling techniques such as SAMMON mapping (Sammon 1969) suffer from the same problem and can be computationally prohibitive. Engineering design applications demand that a design and the tolerances for the design are given in variable space, or at least related to the original variable space, so it is important that any clustering technique relates the clusters found to the original variables.

There is a delicate trade off to be found between providing complex clustering method that partitions data as required (if such an algorithm exists) and the speed of the method as well as understandability and loss of involvement by the user. The latter point is particularly important as data can be clustered in different ways depending on the task in hand; users with different requirements will interpret the same data in different ways, expert users would like to try specific algorithms and change parameter settings if required. However it cannot be assumed that clustering experts with knowledge of all algorithms will be using the system. A general clustering algorithm that returns the main clusters whose definitions can be modified would be more useful for novice users. Details of

parameter settings and how the clustering results were derived should be available for those interested.

Statistical analysis techniques such as PCA will quickly return a specific projection of a data set but alternative projections may offer more interesting partitions in the data as demonstrated by GGobi (Cook *et al.* 1993, 1995) and could be exploited using ICA. Again the clusters returned by projection techniques need to be presented to the user automatically in a fast way. Clustering for engineering design applications also needs to take into account considerations such as the fitness of the design or give more importance to the objective values than the variable values. Therefore it is proposed that fast clustering is performed in the original variables or an alternative coordinate system (such as PCA and ICA), but allow the user to edit the results and relate the results to the original variables. A fast clustering method based on kernel density estimation adapted for engineering design data is proposed in Chapter 4.

Guidelines from the human computer interaction literature suggest supplying an overview and allowing zooming and details of individual data points if required (Shneiderman 1998), in engineering design scenarios the ability to generate new data and delete unneeded data would also be advantageous. The interface should be kept simple but flexible enough to be able to manipulate and analyse complex data (Hutchins *et al.* 1986). More sophisticated commands can be made available to remove frustration for knowledgeable users as it is important to keep the needs of the user at the centre of interface design (Norman 1986, 1998). Intensity of colour, controlled by luminance or brightness and saturation, is the best attribute of colour to use to emulate continuous change, but the number of colours that can be used is limited especially for colour deficient users. Colour would be particularly useful to highlight the important clusters in the data returned by the clustering algorithm or defined by the user, helping to improve perceptual

understanding of the search space. The relative fitness of the solution (in terms of an objective or combination of objectives) can be represented by the brightness of the data item, so that two pieces of information are provided by the colour. The relative merits of regions of the search space in terms of fitness (engineering performance) and robustness can then be assessed visually, as well as using some statistical measurements of those regions.

The mix of visualisation techniques such as parallel coordinates and scatterplot matrices all kept consistent by the linking of colours can provide a rich combination of views that helps understand multidimensional data as experienced by Tweedie *et al.* (1996b). During the interaction process new data will be generated using the genetic algorithm (GA) providing new views, these views need to relate to the original views but have the option to be discarded if necessary. For an interactive engineering system this review recommends combining standard visualisation techniques with advanced clustering procedures, an easy to use interface should provide flexible interaction with a GA for the generation of data as required by the user. Such an architecture will support the iterative process of engineering design (Eckert *et al.* 1999, Dym 1994, Parmee *et al.* 2000, Schön & Wiggins 1992) and is described in detail in Chapter 5.

Chapter 4: Discovering and Characterising Possible Regions of Interest

4.1 Introduction

The review of the literature given in the last two chapters identified that a technique is needed to extract the kind of information a user will require during engineering design tasks. To find this information a number of techniques need to be combined in a novel way. This chapter describes an iterative technique that identifies possible regions of interest that a user may wish to investigate further; regions of interest are characterised by:

1. **High density**. If a genetic algorithm or other optimisation routine produces the data, regions of high density will usually imply a converged state of the algorithm, and thus good solutions.
2. **High fitness.** In some cases high fitness solutions may be found in a region of low density. This may indicate a non-robust or sensitive region of the search space, but the user should be given the opportunity to investigate the region further.
3. **Large relative width of region.** Increased width implies higher robustness, so an idea of the relative size of the region investigated would be useful to the designer.

The algorithm should also have the following characteristics:

1. **Fast.** The speed of the algorithm should not slow down significantly due to the number of points or number of variables in the data. This means that sampling the data may be necessary.
2. **Non-parametric or very few parameters.** To avoid confusion and over-complication, it would be preferable that the user does not have to customise the parameters of the algorithm for each problem encountered. If parameters are

present an automatic parameter selection routine would be preferable. However more knowledgeable users may like to have some control over parameters if possible.

3. **Reliable/robust results.** The algorithm should be predictable in that expected results will be returned and repeated executions will give the same answers. The fact that as the number of dimensions increases the number of solutions required to accurately reflect reality increases also needs to be taken into account.
4. **Suggest other regions of interest.** As well as finding the main clusters in the data for analysis, the algorithm should also suggest further regions that may be worth investigation.
5. **Easy to visualise and understand results.** The output of the algorithm should be easy to understand and interpret. Visualisation of high dimensional output is needed in any coordinate system, but it should be possible to relate the output to the original coordinate system.

The algorithm is based on kernel density estimation (KDE) and can be applied in any coordinate system, the next section explains the idea using the Iris Data and by comparing with the clustering results from Chapter 3. The full algorithm is then described and demonstrated on high dimensional test functions showing how clusters can be identified in any coordinate system and related to the original variables.

4.2 Application of Kernel Density Estimation (KDE) to the Iris Data

In Sections 3.4.3 and 3.4.4 it was shown that principal component analysis and independent component analysis can find interesting projections in data that could be used to cluster and classify different regions of the data. However in that section, in a similar way to most literature, the classification was done visually. The aim of this system is to automatically identify the main clusters in the data and suggest other regions to look into.

Univariate kernel density estimation is a way of analysing the projected data along each vector of the search space, either from the point of view of the original variables or natural variables such as the principal or independent components. Univariate kernel density estimation gives an idea of the maximal and minimal regions of density along each dimension. A multidimensional cluster can be deduced by integrating the univariate information from each dimension.

Kernel density estimation (KDE) could be described as a smooth version of the histogram, see Silverman (1986) for an excellent overview and Beardah & Baxter (1996) for a useful MATLAB toolbox. Consider a univariate data set x with true density function f and a set of n arbitrary points X_i chosen along the range of x. The kernel K is a smooth function that is placed over each point X_i, then each original data point in the vicinity of X_i is given a weight according to K. The kernel function is usually a symmetric probability density function with well-behaved properties and definitely satisfies the condition:

$$\int_{-\infty}^{\infty} K(x)dx = 1 . \qquad \textit{Equation 4.1}$$

The density estimate is calculated by summing all the individual kernels using a window width h, also known as the smoothing parameter, producing a smooth density estimate of the original data x:

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right) \qquad \textit{Equation 4.2}$$

where x = the data set

X = an arbitrary subset of points along the range of x

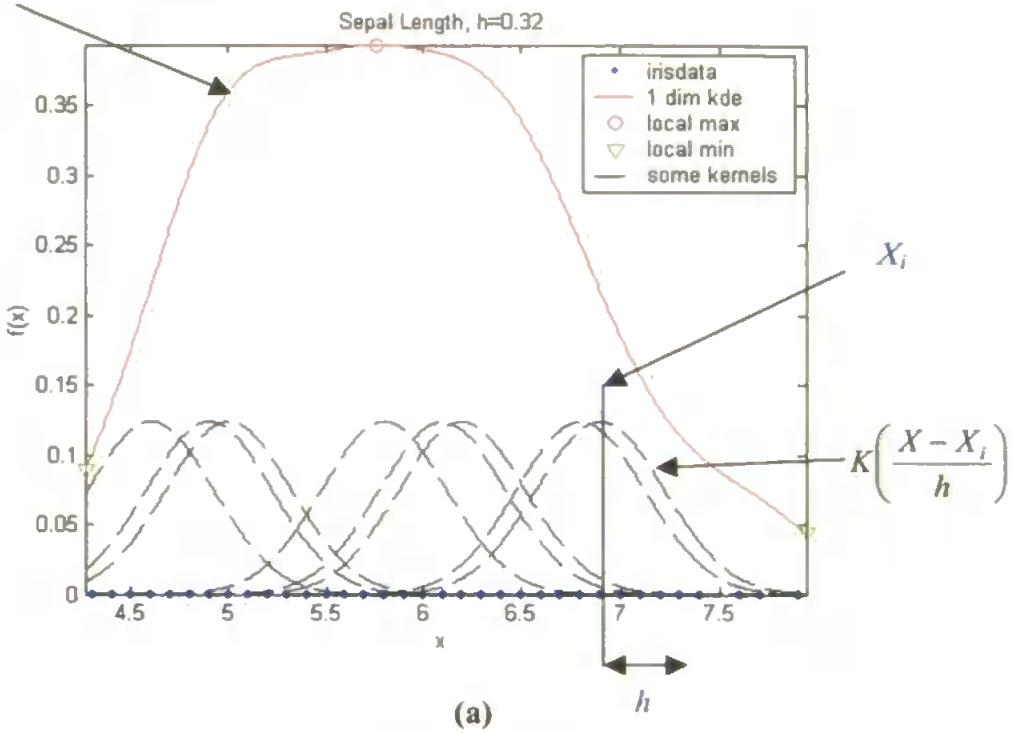
n = the number of data points in X

K = a kernel function

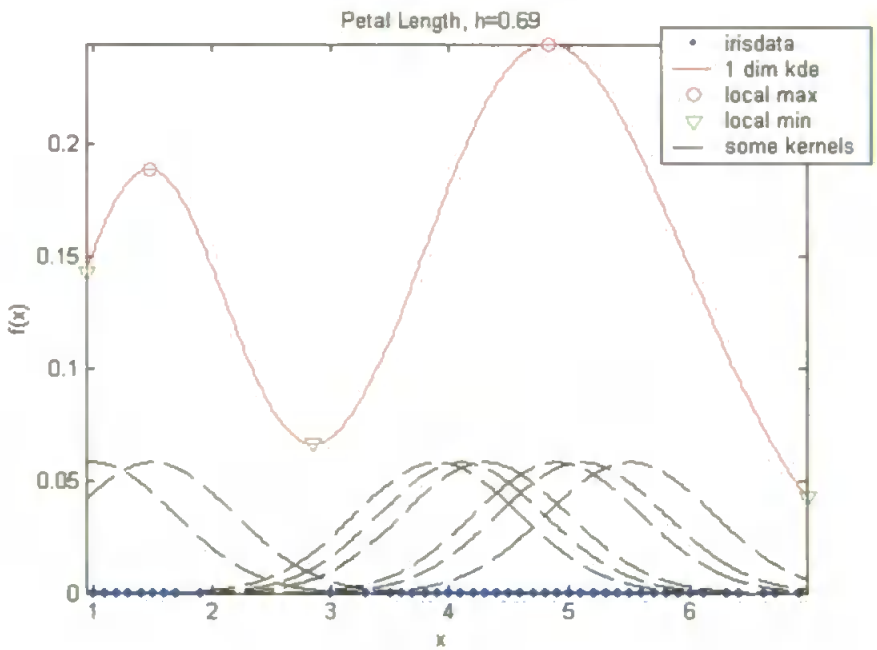
h = the window width or smoothing parameter

$\hat{f}$ = the density estimate.

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right)$$



(a)



(b)

Figure 4.1: Two examples of univariate kernel density estimation (KDE) taken from the Iris Data. Some examples of the individual kernels are shown as dashed lines, the sum of these kernels make up the solid red density estimate *f*(*x*). The minimum and maximum of this line can be computed to give an indication of the position of clusters in each dimension. Note the different values of *h* in (a) and (b) because they are based on the standard deviation of the data. The kernel density estimate is found using a MATLAB toolbox (Beardah & Baxter 1996). Used with permission.

Figure 4.1 shows a graphical output of the density estimate on two variables of the Iris Data. Choosing the kernel K and the value h are critical in changing the shape of the density estimate. The choice of h is the most important part of the density estimate, as it will strongly influence the result of the estimate. An optimal value of h is derived by minimising the mean-squared-error of the density estimate from the actual density; unfortunately this optimal value still depends on knowing the actual density. However the optimal kernel can be deduced by calculus and is called the Epanechnikov kernel. The well-known normal or Gaussian kernel is similar to the Epanechnikov; it is not as efficient but is often used because it is common practice to assume data has a normal distribution in the first instance. For normally distributed data with true variance σ^2 the optimal smoothing parameter for the Gaussian kernel is given by:

$$h_{opt} = 1.06\sigma n^{-1/5}. \qquad \textit{Equation 4.3}$$

Thus the sample standard deviation could be used to estimate σ. But if the data is multimodal this estimate will oversmooth the data and Silverman (1986, p. 47) suggests a more suitable estimator should replace σ.

There are multivariate versions of KDE (Silverman 1986; Scott 1992) that could be applied to high dimensional data generated by engineering design applications. However the research into multivariate density estimation is limited, the calculation of h becomes more difficult and the computation time slows down. The other option is to use the univariate estimates of h and combine the density estimate from individual variables in some way to cluster the data (Beardah 1999). The latter option was more desirable for engineering design data for two main reasons; firstly the univariate density estimation is a lot faster than multivariate, secondly the reason for using density estimation was to identify regions of high density using the local maxima and minima (this is difficult to do when

considering multivariate estimates because the density may fall away at different rates in different directions). So this alternative strategy was chosen: multivariate clusters are formed by integrating the information from the local minima and maxima of univariate kernel density estimates as described in the following paragraph. Once this decision was taken it became apparent that oversmoothing the univariate density estimate was desirable since the integration of univariate estimates usually results in smaller clusters. Investigation with various kernels showed that the Gaussian kernel returns more general clusters than the more accurate Epanechnikov kernel and the value of h_{opt} given previously with the sample standard deviation used to estimate σ were chosen to achieve a large value for h. These parameters are the default settings for the density estimates used in most of the examples given in this thesis.

Figure 4.2 shows the idea working on the four dimensions of the Iris Data. The univariate density estimate is computed for each variable. The local minima and maxima of this estimate are trivial to compute by listing the densities in order and comparing neighbouring values. It can be seen that there is just one maximum in the upper two plots of Figure 4.2a – the Sepal Length and Width variables (var 1 and var 2), but the Petal Length and Width variables show two distinctive maxima at around 4.8 (var 3) and 1.5 (var 4) respectively. The bounds of the cluster corresponding to the largest density are defined by locating the minima either side of the global maximum in each variable (the lower bounds are identified by the green triangles at variable values {4.3, 2, 2.8, 0.8}, with the upper bounds at {7.9, 4.4, 7, 2.5}). The first cluster is the subset of data such that each variable value falls inside its corresponding bounds. So clusters are defined by integrating the local maxima and minima information from the density estimate of each variable.

The next cluster is found by temporarily removing the first cluster and performing another iteration of the procedure described in the previous paragraph on the remaining

data. The new analysis in Figure 4.2b shows that nearly all of this data is inside the region defined by the local minima next to the global maximum of the kernel density estimate. So this algorithm has split the *Iris Setosa* class from the rest of the data (Figure 4.2c), except for one data point that may be a recording error in the original data (see variable 2).

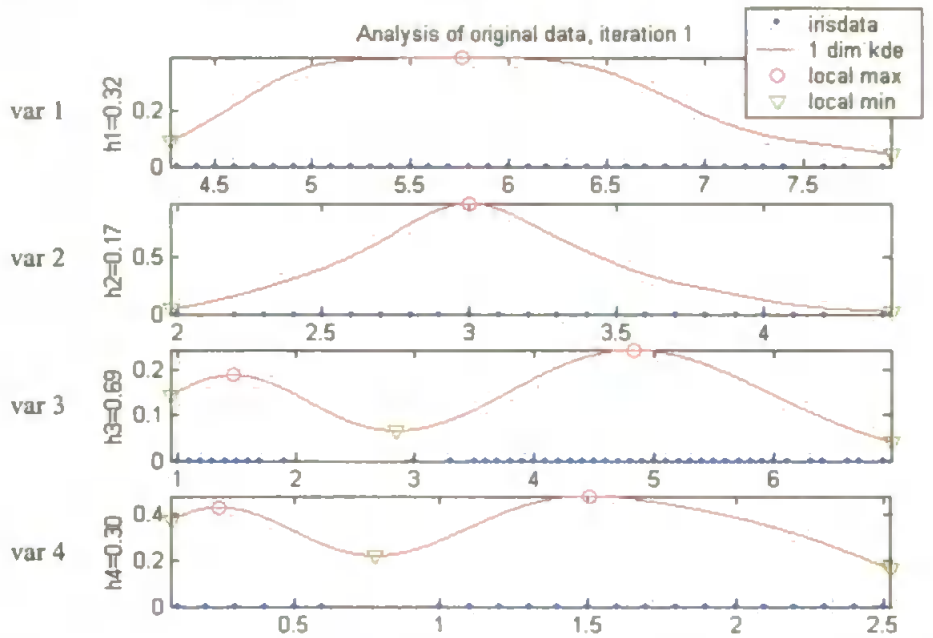


Figure 4.2a: Univariate KDE analysis of the Iris Data. Variables 3 and 4 clearly separate the data. The region of highest density is identified and removed (i.e. from 2.8<var3<7 and 0.8<var4<2.5).

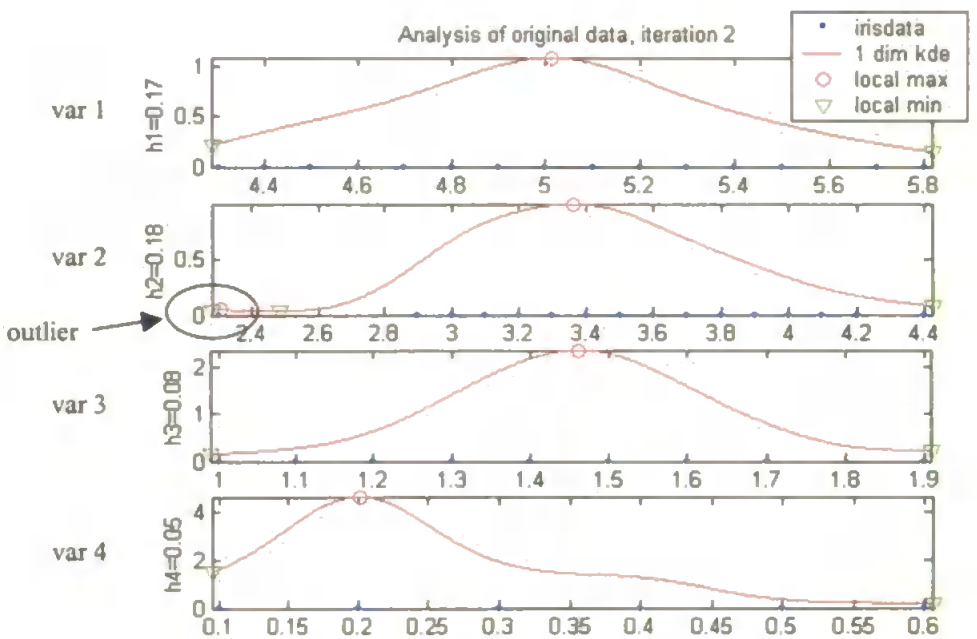


Figure 4.2b: The remaining data is analysed, leaving just one data point (var2=2.3) as unclassified.

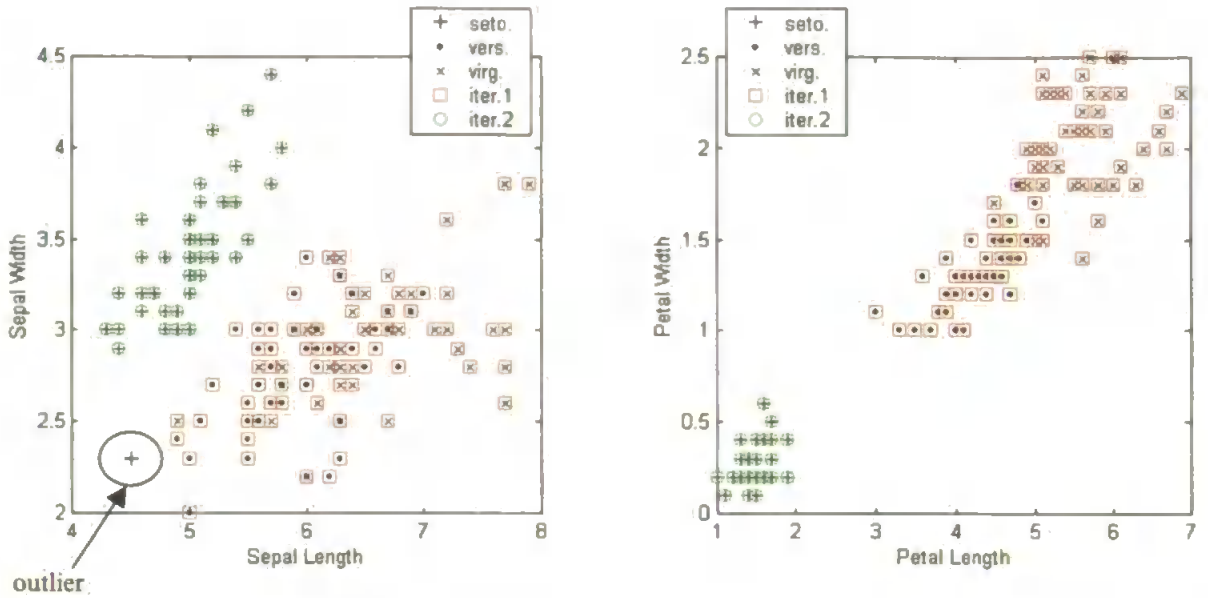


Figure 4.2c: The classification results. The actual classes are *Setosa*(+), *Versicolor*(.) and *Virginica*(x), the algorithm separates the data into two clusters shown by the red squares and the green circles which are nearly all *Iris Setosa*.

Univariate KDE can be performed on any vector through the data. For example using principal components analysis (PCA) reveals a similar result as seen in Figure 4.3. The first principal component clearly separates the *Iris Setosa* class from the rest of the data (Figures 4.3a and 4.3b). The result after two iterations shown in the original variables (Figure 4.3c) is similar to Figure 4.2c but this time the outlying point is classified correctly as *Iris Setosa*. Separation in the first principal component is sufficient to identify two clusters in the Iris Data. This feat should be achieved by any linear clustering method, indeed *k*-means and shared nearest neighbour clustering (see Sections 3.5.3 and 3.5.4) achieved this and were partly successfully in separating the other two classes.

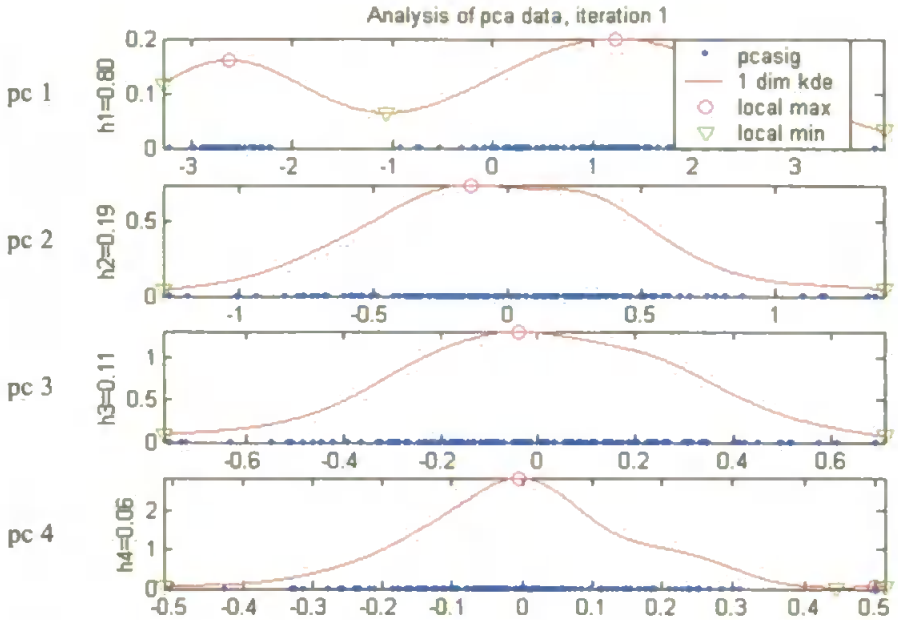


Figure 4.3a: Univariate KDE analysis on the principal components of the Iris Data gives similar results.

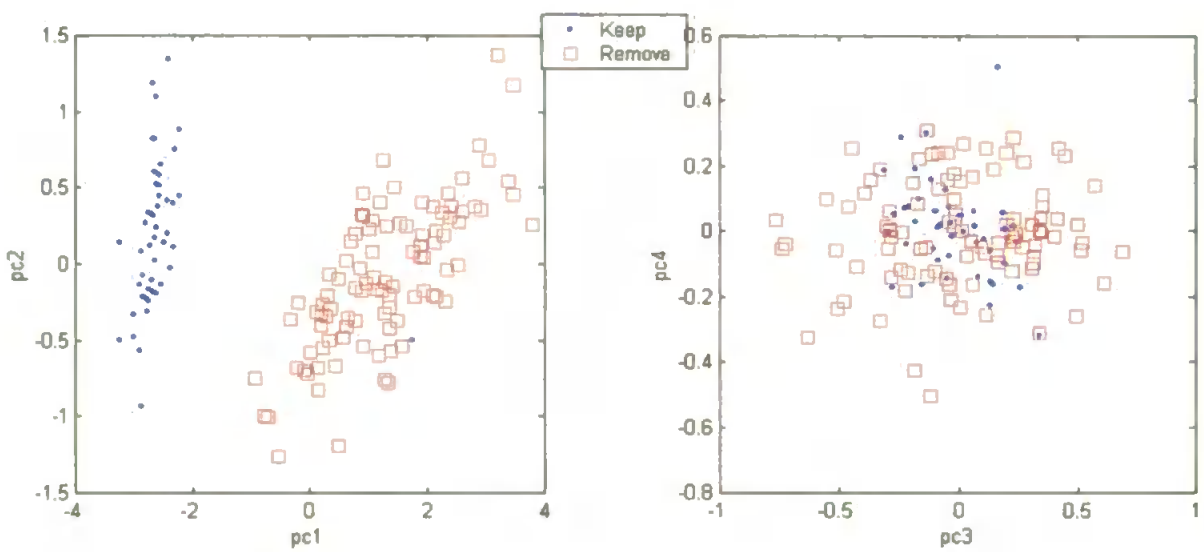


Figure 4.3b: The first principal component easily separates the Iris Data.

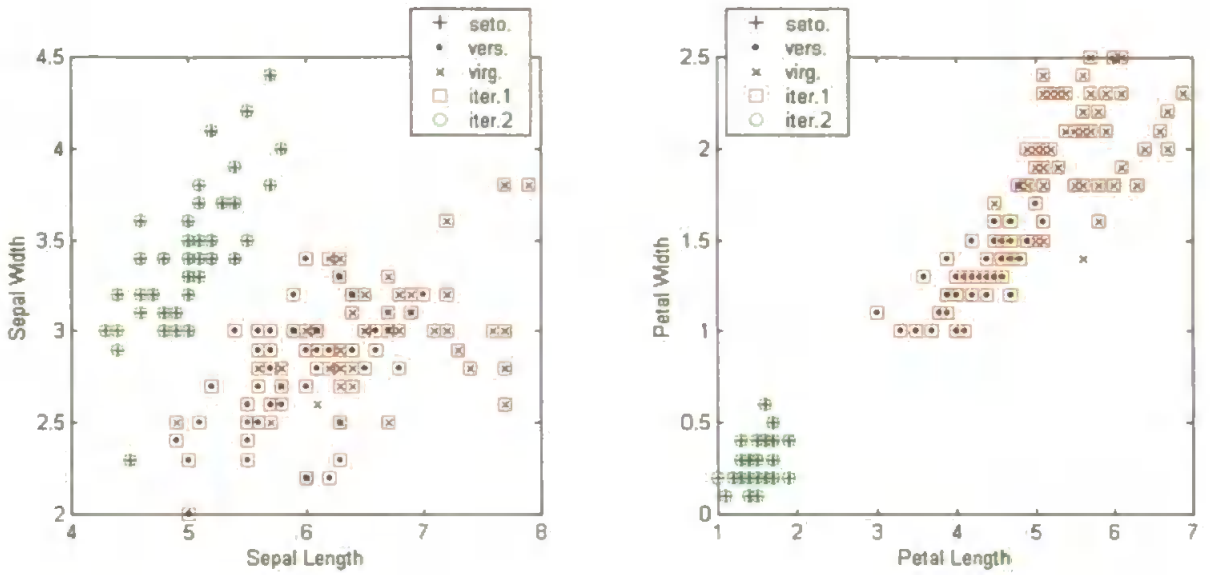


Figure 4.3c: Clustering using PCA and KDE, very similar to the results from the original variables in Figure 4.2c.

Independent components analysis (ICA) of the whole Iris Data gives a similar result. However if the user is interested in doing a further analysis of the main cluster, then the independent components give an interesting result. Figure 4.4 shows analysis of the Iris Data with *Setosa* removed. The first independent component (searching for "nongaussianity") indicates a separation in the data (Figure 4.4a top plot and Figure 4.4b). The next iteration clusters most of the rest of the data together. When the chosen clusters are mapped back to the original variables it is seen that they correspond to the *Virginica* and *Versicolor* clusters up to around five exceptions. This result is better than the results given by the clustering algorithms in Section 3.5, albeit working on the data with *Iris Setosa* removed.

These results demonstrate that clustering with univariate density estimation does locate the main clusters in the data and can identify subtle clusters using certain coordinate systems with results that are no worse than partitional clustering algorithms. The speed of the density estimate is very fast and does not require any expensive calculation of the

distance matrix between all solutions in the data set. To identify useful clusters in engineering design data, the fitness or objective values of the data need to be taken into account, the next section shows how this clustering method can be easily adapted to include such information.

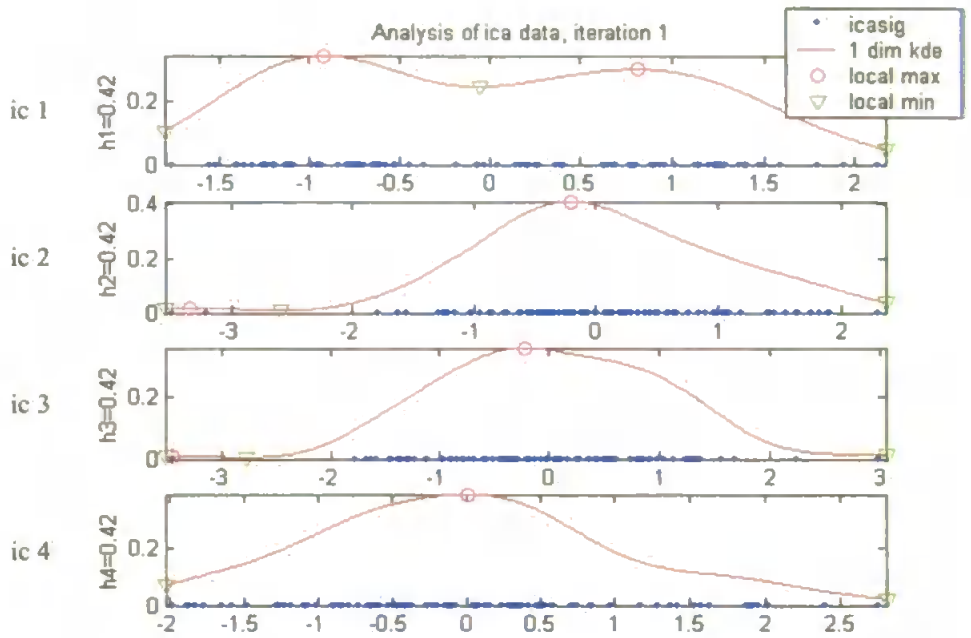


Figure 4.4a: Univariate KDE analysis of ICA working on the Iris Data with *Setosa* removed. Here clear separation is identified in the first independent component (ic1).

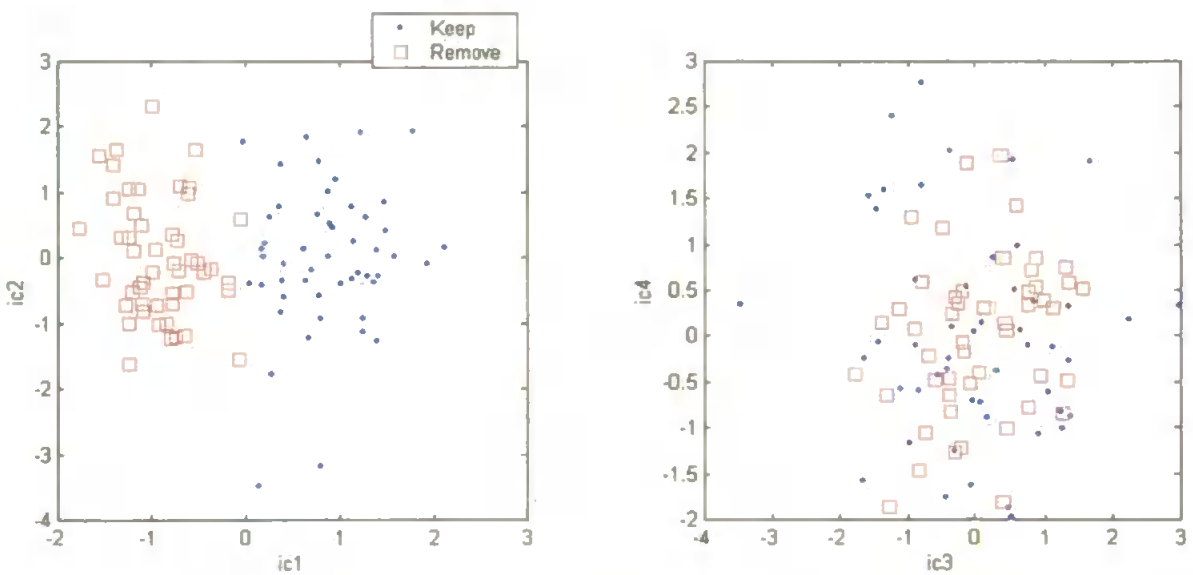


Figure 4.4b: Solutions identified from iteration 1 using the independent components.

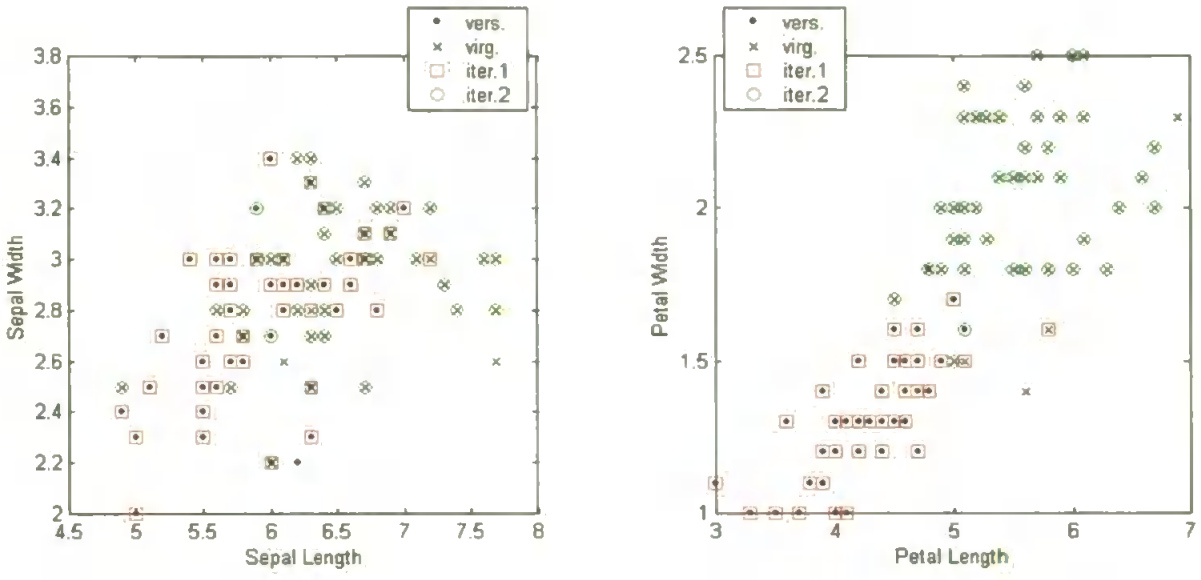


Figure 4.4c: The clustering results using ICA on the Iris Data, separating *Virginica* from *Versicolor* up to a few exceptions (some data is not classified at all).

4.3 A Clustering Algorithm based on Kernel Density Estimation

4.3.1 Engineering Design and Genetic Algorithm Considerations

The univariate KDE method finds natural clusters in the data, but for engineering applications the data also has objective values or a fitness value associated with each data point. The location of high fitness information is important; if it is inside the region of high density then the user can be confident of looking in a good region. If high fitness is outside the region of high density then further investigation is required into the high fitness region. From an engineering design point of view the regions containing the fittest points are most interesting at first; after examination the user may wish to study other regions of high density with relatively low fitness, but by default the algorithm will identify the cluster containing the highest fitness at each iteration.

When a genetic algorithm (GA) converges, even over a short run, many copies of individuals will be created and saved. If this is ignored the density in converged regions will be even higher due to repeated data. One solution would be to remove the repeated

data before analysis. However it would be more efficient not to create the same data during the GA. Therefore any solution that (after selection, crossover and mutation) has the same genotype as one saved previously, will be further mutated until a unique genotype occurs. For a relatively low number of generations and chromosome length this approach can be implemented at negligible computational cost compared to the calculation of a real-world engineering function. The advantage of this method, over sharing and crowding for example, is that no parameters are required. The new solution is still related to the original so the evolutionary concept is not lost. However, if the chromosome length and number of generations is very large the savings on computation cost may not be so favourable.

Another question that arises during the design of the algorithm is whether the univariate KDE should be performed on the original variables, the principal components or independent components. The advantage of using independent components when applied to the Iris Data was shown in the previous section. However for engineering design data the advantages are not so clear. The reasons are best explained by a visual example that will be given in Section 4.4 after the algorithm has been explained in a step by step example.

4.3.2 The Algorithm with Example

To show the idea on data where fitness is involved, the modified Himmelblau function will be used. Beasley *et al.* (1993) used a version of the Himmelblau function to test multimodal optimisation techniques. Here the Himmelblau function is modified so that the function is maximised and the global maximum is zero. The two-dimensional version is given by:

$$f(x_1, x_2) = -(x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2 \qquad \textit{Equation 4.4}$$

A surface plot of the function is shown in Figure 4.5 in its usual domain: $-5 < x_1, x_2 < 5$. In this case the fitness or objective function is given by f. There are four optimal regions of similar fitness but one is more sensitive and isolated than the others, so the GA is less likely to converge on that region. This function is more interesting in four dimensions as it has 16 local optima and is more difficult to visualise and understand:

$$f(x_1,x_2,x_3,x_4) = -(x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2 - (x_3^2 + x_4 - 11)^2 - (x_3 + x_4^2 - 7)^2$$

Equation 4.5

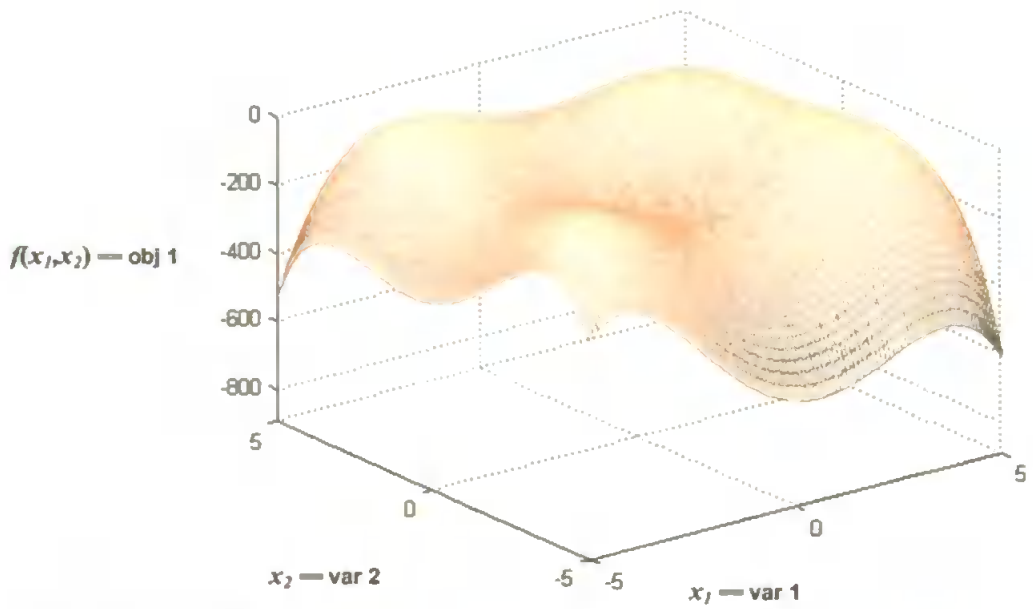


Figure 4.5: The modified Himmelblau function – two dimensions.

From this point onwards a notation change is used, reflecting the assumption that the data is of an engineering rather than mathematical nature: x_1, x_2, x_3, x_4 are known as variables – hence renamed 'var 1', 'var 2', 'var 3', 'var 4'; the objective value $f(x_1,x_2,...)$ is renamed 'obj 1' (see Figures 4.5 and 4.7). For some engineering problems there may be a number of objectives, in this case the objectives will need to be combined in some way to form the fitness. For simplicity one objective is assumed and is interchangeable with the fitness, so a solution to the four dimensional Himmelblau problem is therefore made up of the values 'var 1', 'var 2', 'var 3', 'var 4' and 'obj 1'. This assumption is also made in the description of the clustering algorithm (Figure 4.6): the fitness f is the same as a single objective 'obj 1'.

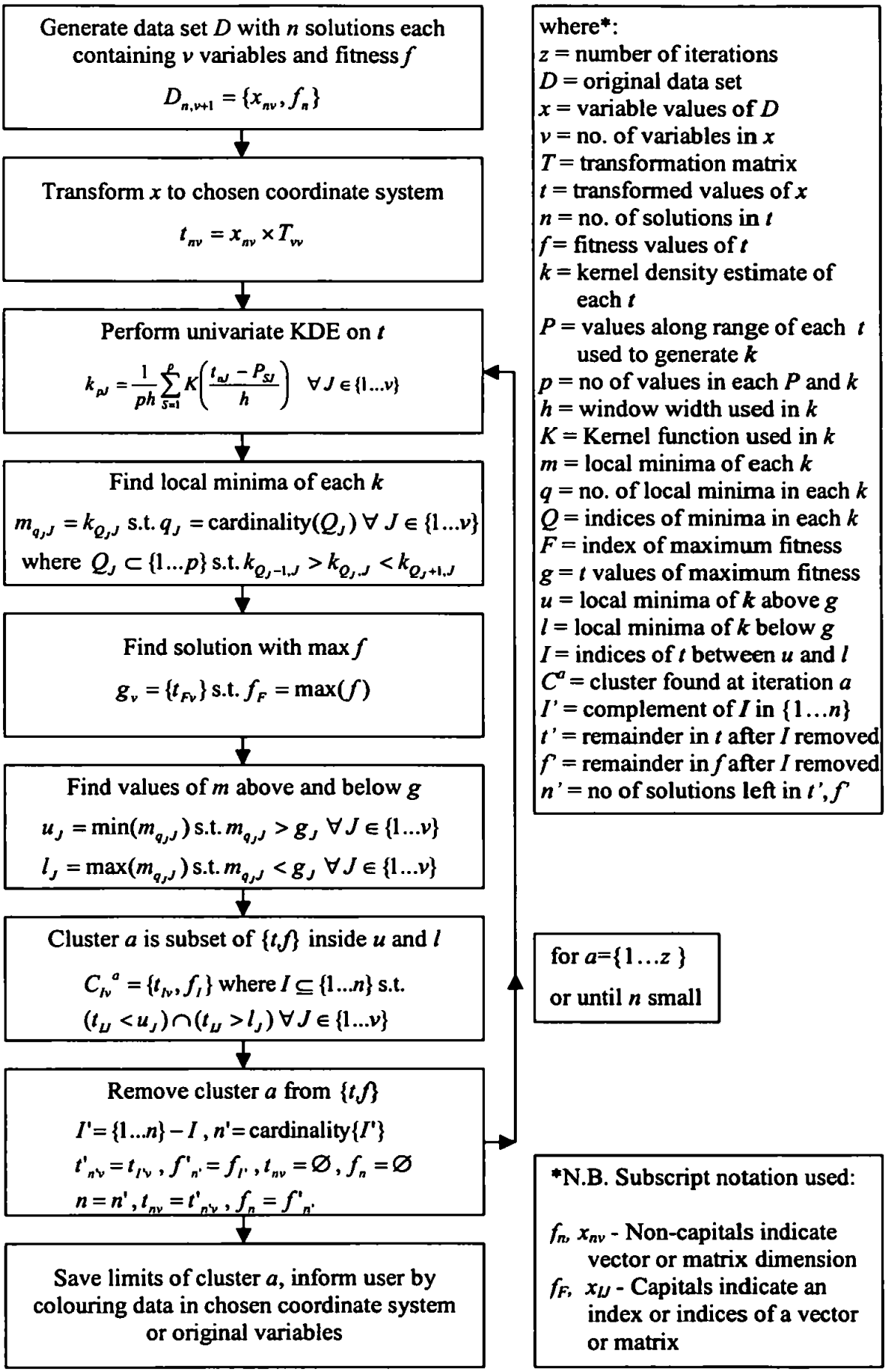


Figure 4.6: The univariate KDE-based clustering algorithm for data including objective value information (assuming 1 objective is the same as the fitness).

The clustering algorithm based on KDE for data generated by a genetic algorithm is summarised in Figure 4.6 whilst a practical example is shown in Figure 4.7. Figure 4.7a shows some data generated by a simple genetic algorithm running on the four dimensional Himmelblau function for twenty generations using the mutation of repeated chromosomes (as described in Section 4.3.1). Assuming that clustering is to be performed in the original variables, so no transformation of the variables is required, the result of the univariate KDE analysis is shown in Figure 4.7b. The solution with maximum fitness is shown as a red circle in each variable. The local minima of the density estimate either side of the maximum fitness are at approximately the middle and top range of each variable, this is also the region of maximum density. This region is chosen as the first interesting region and any data falling inside the limits of all variables is defined as the first cluster. Figure 4.7c highlights this data as red squares.

Once the region has been defined, that data is removed and the second iteration of the algorithm continues on the remaining data. KDE analysis of the second iteration (Figure 4.7d) is similar to the first, but this time the lower range of variable 2 contains the maximum fitness. Again the data inside the limits identified by the local minima is defined as cluster 2 and removed from the data set (large green circles in Figure 4.7e). The third iteration shows that in some cases the maximum fitness solution is not contained in the region of maximum density (Figure 4.7f, 'var 2'). This fact can be brought to the user's attention, but in the current implementation the region containing the maximum fitness is chosen. Figure 4.7g shows the clustering results on all the data after four iterations, four clusters (red, green, blue and yellow) have been found indicating four distinct local optima. The user could choose to find more clusters in the same way.

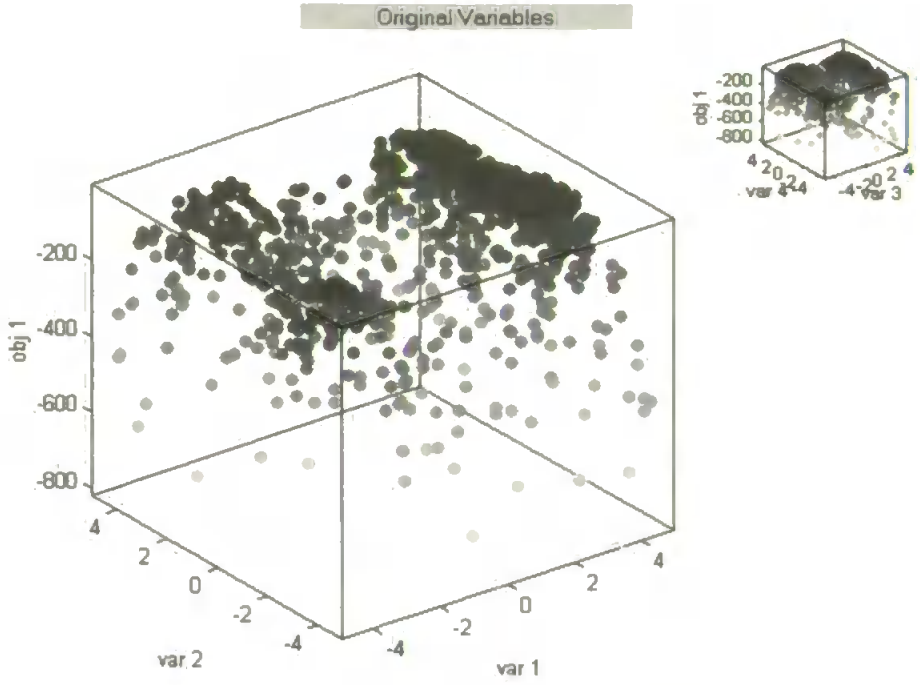


Figure 4.7a: Result of genetic algorithm: 100 individuals, 20 generations, so 2100 unique data points (including original random individuals). Note variable names have changed to reflect engineering design scenario: 'var 1'=x_1 ... 'var 4' = x_4, 'obj 1'=f.

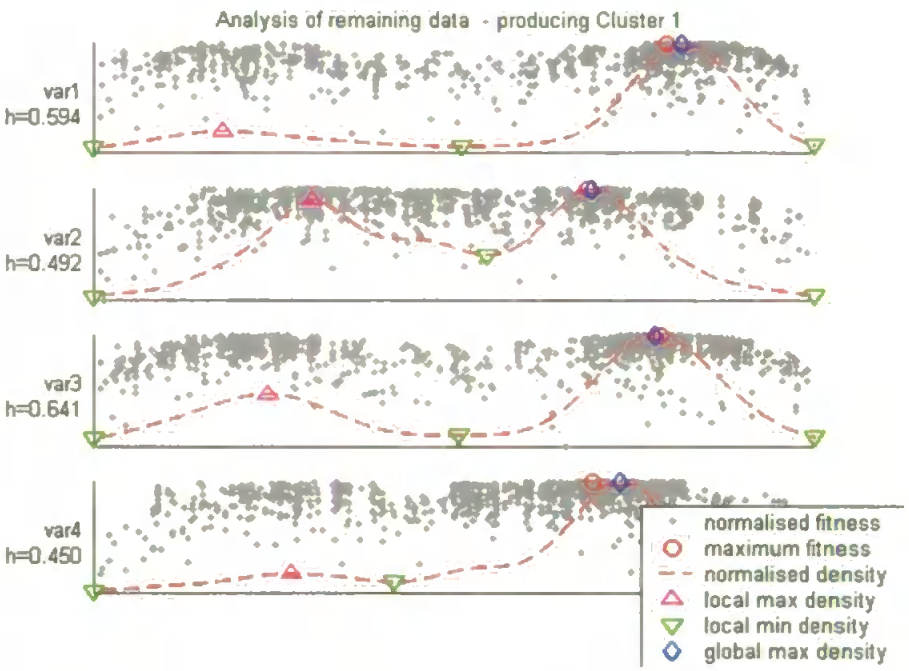


Figure 4.7b: Density analysis of Himmelblau data. The dashed red line is the density estimate, the green downward pointing triangles are local minima and the blue diamond represents the global maximum of the density estimate. The black dots indicate the relative fitness of the data points (normalised and scaled to compare with the density information). The maximum fitness point shown as a red circle, in this case it is inside the region of maximum density.

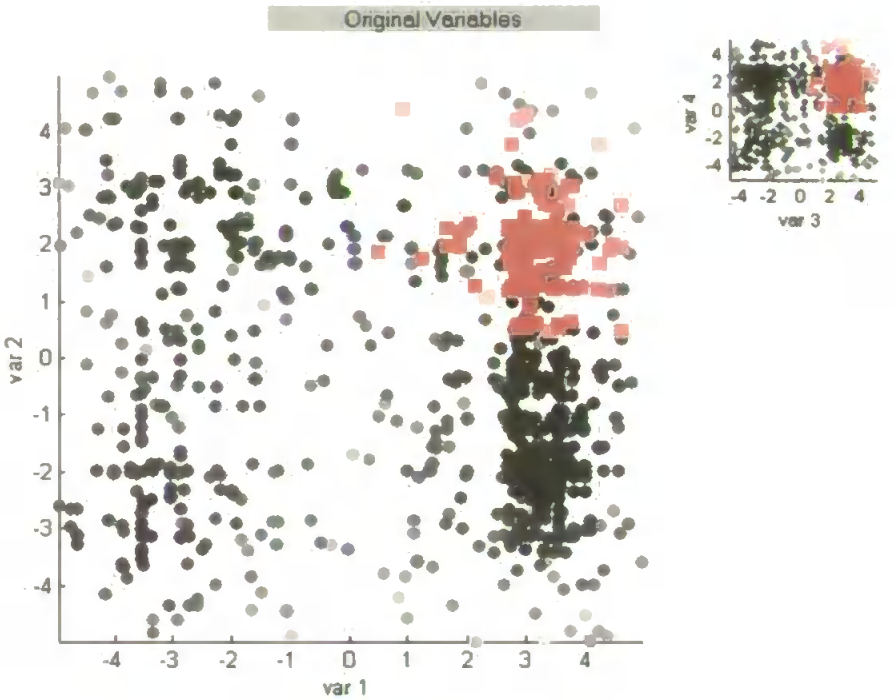


Figure 4.7c: Red squares depict the cluster identified by the first iteration of the algorithm, only data that falls inside the upper half of every variable is included.

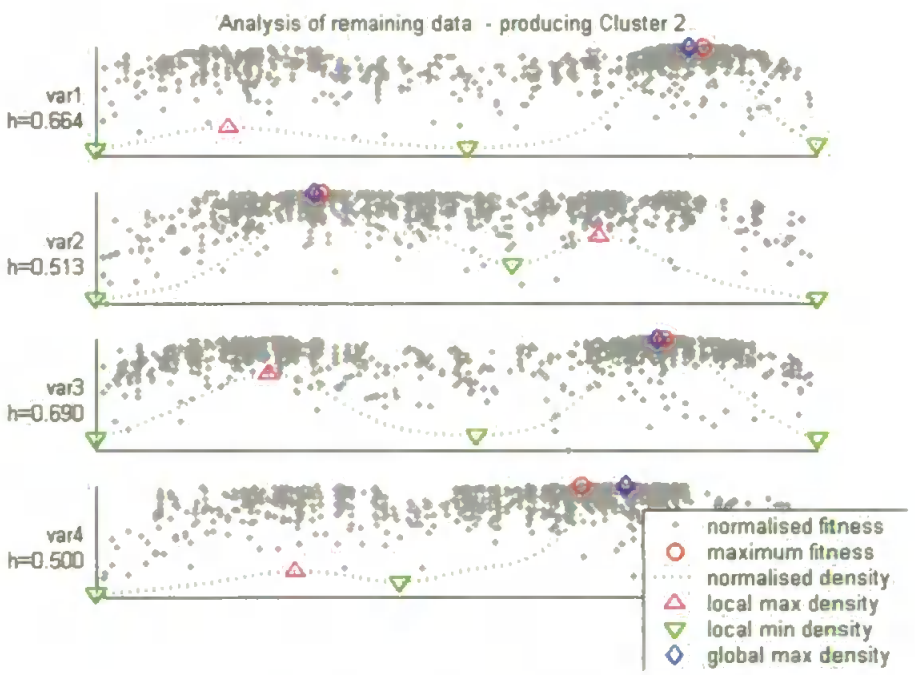


Figure 4.7d: Iteration 2 – analysis of data with cluster found in iteration 1 removed, region containing maximum fitness the same as region of maximum density again.

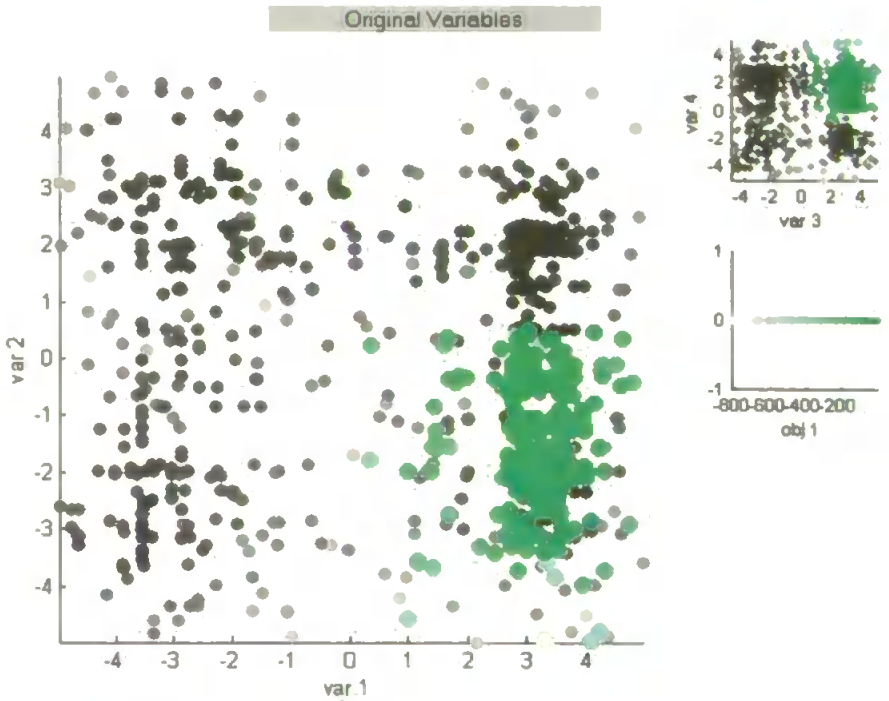


Figure 4.7e: Second region to be removed, this shows that only 'var 2' splits the first and second cluster.

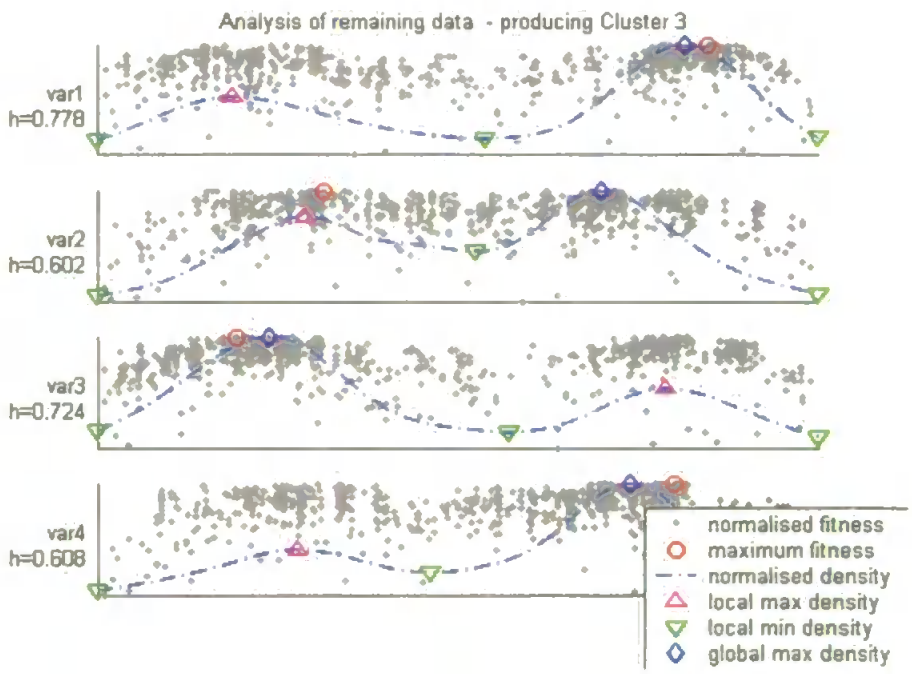


Figure 4.7f: Result of the third iteration. Maximum fitness (red circle is not contained in the region of maximum density (blue diamond); separated by variable 2.

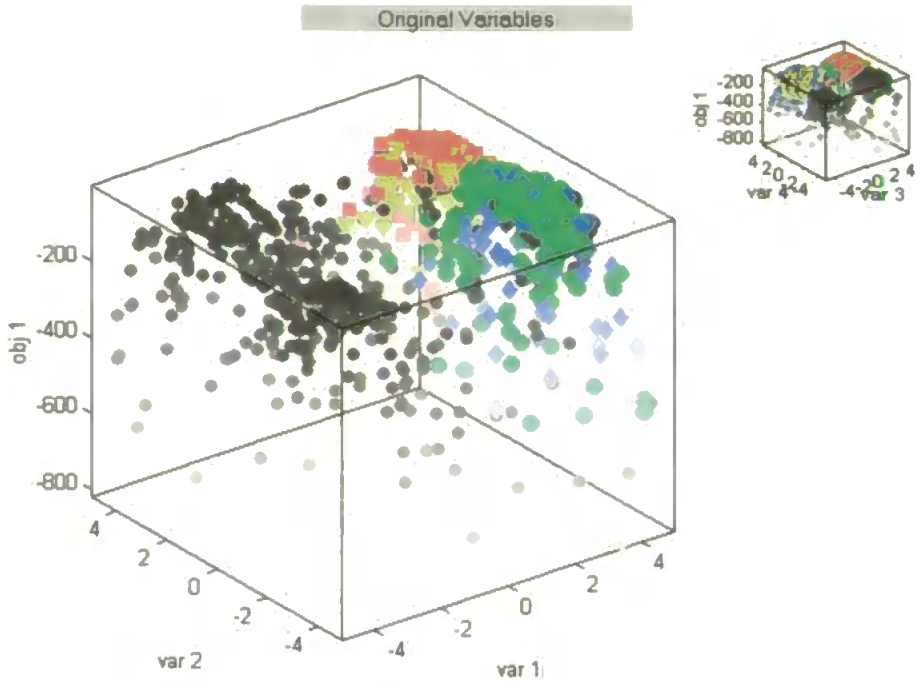


Figure 4.7g: Result after four iterations, four distinct clusters corresponding to four local optima; there are sixteen optima present in this function.

As the algorithm is iterative there is so no particular starting or stopping criteria. Partitional clustering algorithms require the number of clusters to be given or found by repeated iteration of the algorithm and minimising of some error function (Sierra & Corbacho 2000). At each iteration the kernel density estimate will return a fixed cluster, so the algorithm can continue until all the data has been assigned to a cluster. Therefore the user can ask for a certain number of clusters to be found, the algorithm will continue for that number of iterations unless the number of points becomes low (say 10% of the original). The user can then ask to find more clusters if needed, but is more likely to be interested in investigating the clusters already returned.

The relative fitness and size (in terms of hypervolume in variable space) of the clusters can be assessed by the user who may decide whether further investigation is warranted. The size of the cluster may be related to the robustness of the region; if the minimum fitness of solutions in the cluster is not too low then a larger cluster is more

attractive because the manufacturing tolerances will be larger. However a definitive value for robustness cannot be given because the space has only been partially sampled by the genetic algorithm and further search may be required to validate the robustness assumption. For this reason the difference between a region of the search space and a set of solutions found inside it is critical; in this thesis the word 'cluster' refers to a set of data points found, whilst the volume of variable space the cluster encompasses is termed the 'region' of space.

4.4 Application of Alternative Coordinate Systems

One choice that needs to be made before running the clustering algorithm is whether to use the original variables or other, possibly more natural, coordinate systems such as the principal or independent components. It was hypothesised that natural vectors would show up the main clusters in the data because they are being analysed from the point of view of the actual shape of the data. So a 'PCA friendly' function was devised to test this hypothesis. The function has diagonal ridges across the search space so that the 'natural vectors' will be at 45 degrees to the original ones (Figure 4.8). The ridge is formed by a number of peaks along the diagonal using the formula based on functions by Fonseca & Fleming (1993) described in Chapter 6. The same configuration is repeated in two more variables to give a four-dimensional version of this function.

Figure 4.9a shows some data generated by a GA spread out somewhat along each ridge. The univariate KDE analysis of this data in the original variables (Figure 4.9b) shows how the GA converges on certain parts of the search space (in fact two separate GA runs were combined to form this data). The data is partitioned by variables 1 and 3 and partially by variable 2. The clustering algorithm successfully identifies two clusters in this data as shown as red squares and green dots in Figure 4.9c, in these two-dimensional views the diagonal nature of the ridges are clear, note that higher fitness is indicated by a darker

colour (see 'obj 1' values below the 'var 3' versus 'var 4' plot). Some of the data (next to the red cluster) is not clustered because of the partition in variable two. The cluster definitions are of course at 45 degrees to the natural clusters, so do not reflect the clusters as a human would perceive them.

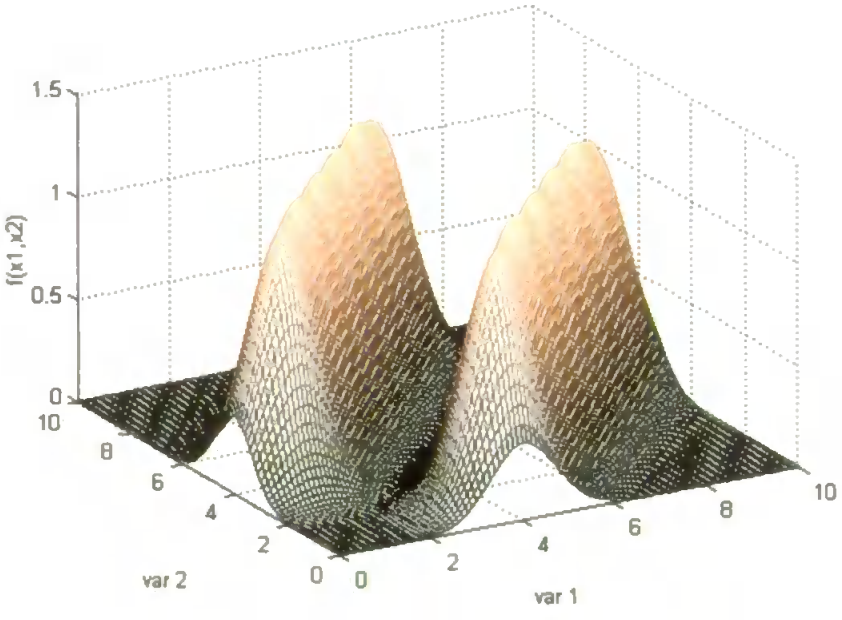


Figure 4.8: Special function that should be advantageous to PCA: wide variance along diagonal.

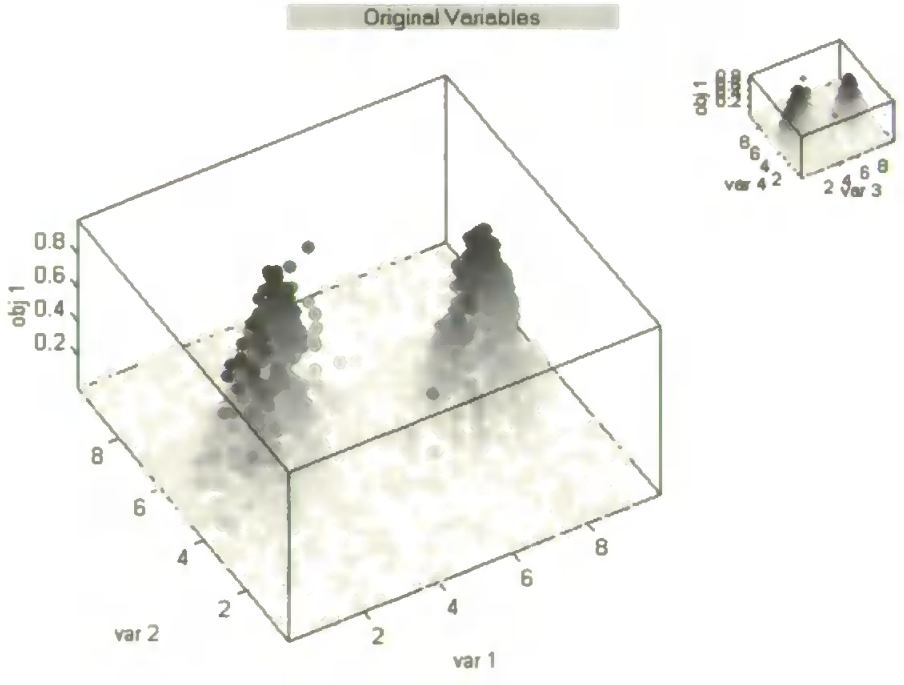


Figure 4.9a: GA generated data in the original variables

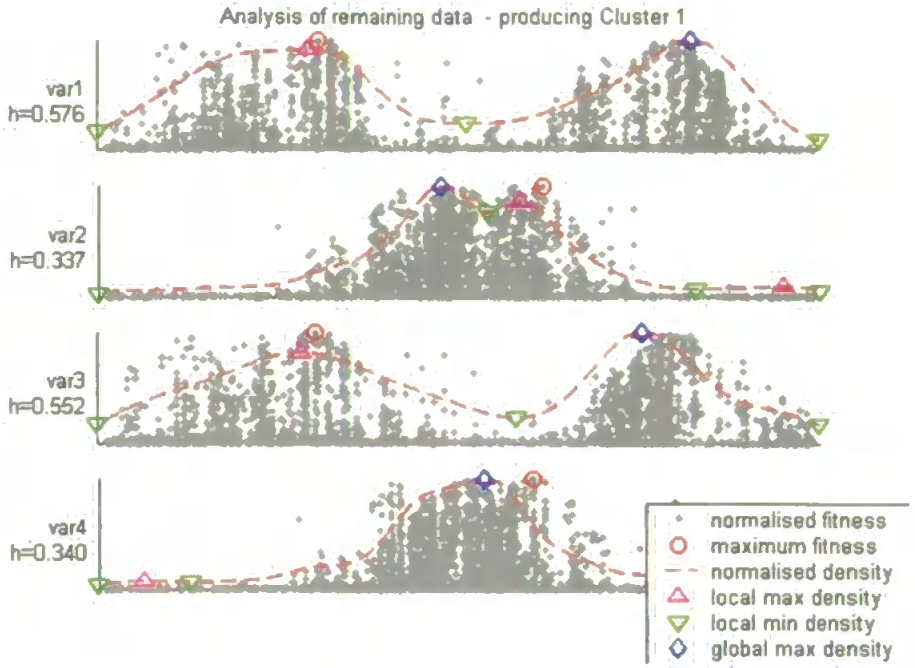


Figure 4.9b: KDE analysis of the original variables.

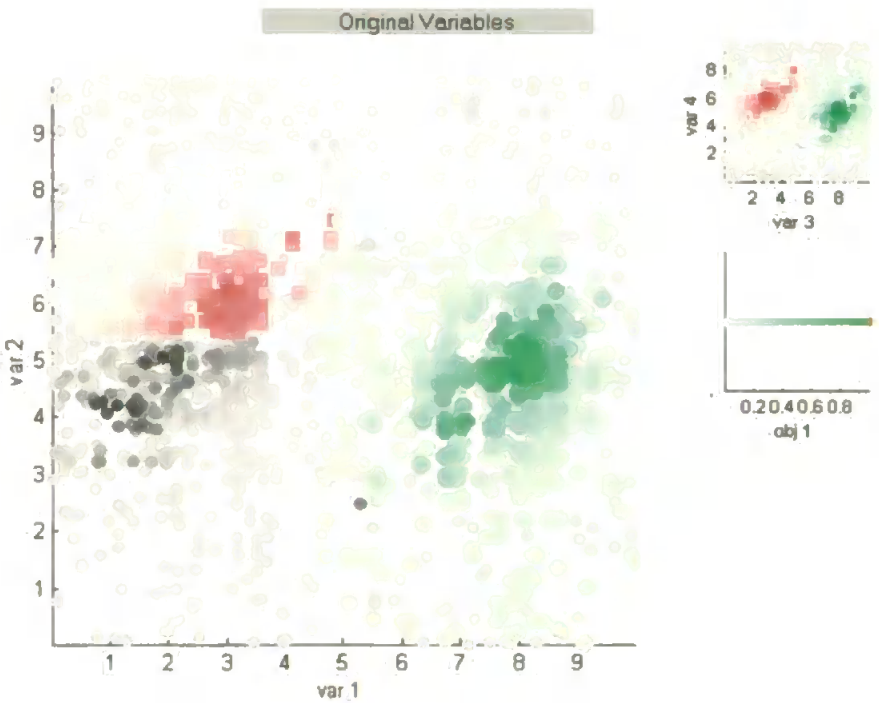


Figure 4.9c: First two clusters found after analysis of the original variables. Both ridges identified, but defined at 45 degrees to perceptual clusters.

Figure 4.10a shows the univariate KDE analysis of the principal components of this data. This time the data is partitioned in the first principal component only, although viewing the result in two dimensions makes the nature of the partition clearer (Figure 4.10b). The clustering algorithm again finds two clusters (blue and yellow). The yellow cluster is drawn after the blue, so may obscures any duplicate information. Visualising these clusters in the original variables shows that all the data has been assigned to one or other of the clusters (Figure 4.10c). The two ridges are mostly defined by one or other cluster, but there is some overlap; in particular low fitness information is assigned to the 'wrong' cluster. This visualisation is more in line with how the user perceives the data, especially for the higher fitness material. The definitions of the cluster limits can be given in terms of the original variables using the transformation matrix (in this case the eigenvectors) and the mean of the data (see Section 3.4.3).

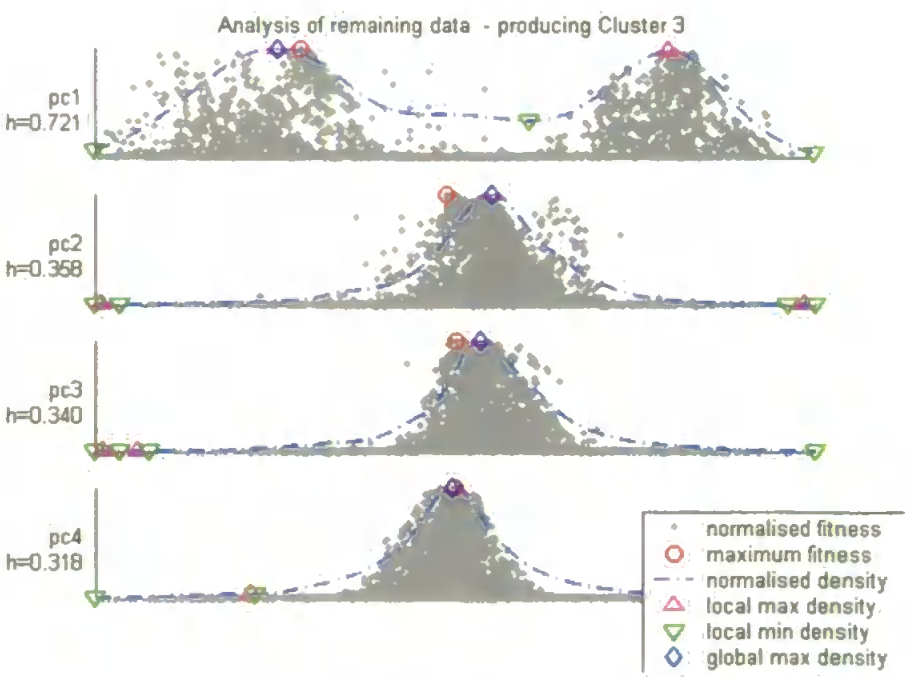


Figure 4.10a: KDE analysis of the principal components. Two peaks identified in first principal component (pc1).

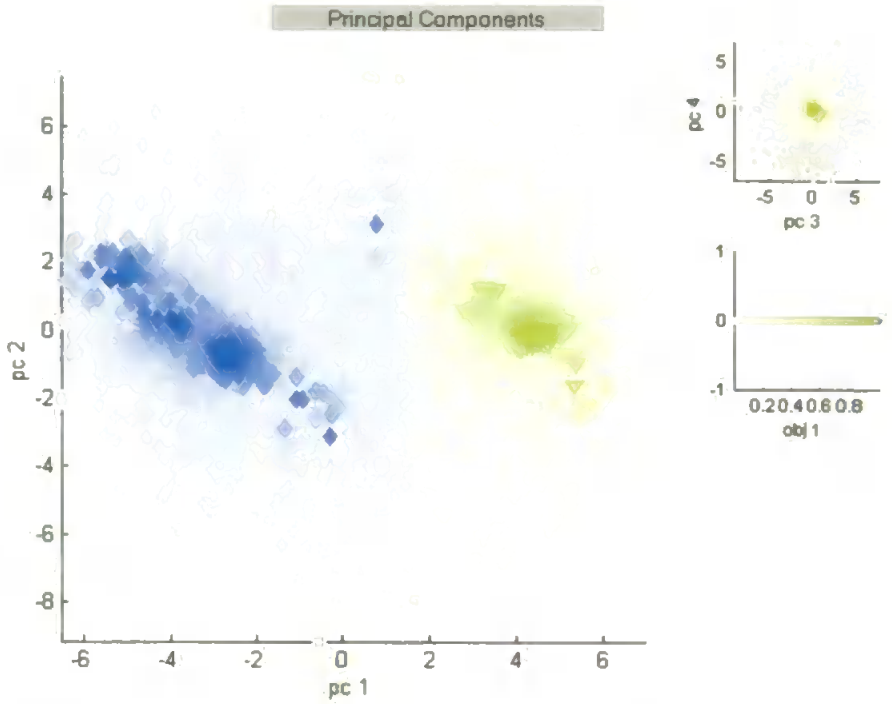


Figure 4.10b: Visualisation of two clusters in the principal components, blue and yellow.

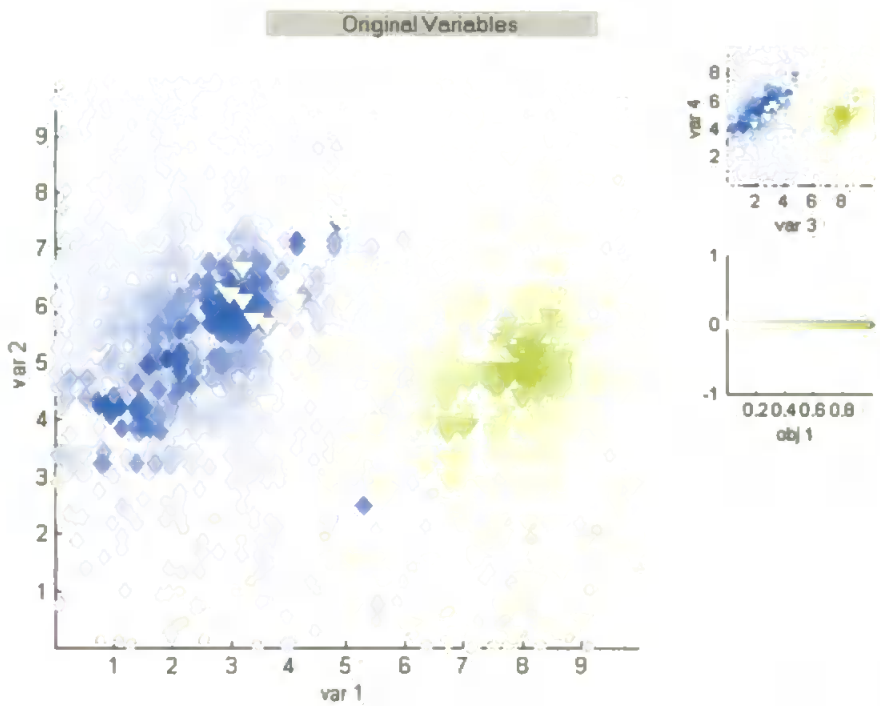


Figure 4.10c: PCA defined clusters shown in the original variables. Both ridges are now identified although there is some unwanted overlap. The yellow cluster is drawn last so may obscure some blue data.

In the same way ICA can be applied to univariate KDE; the analysis shows that most of the data is also partitioned by the first independent component (Figure 4.11a). Some structure is also seen in the third and fourth independent components, which is due to ICA searching for vectors in the data that maximise "nongaussianity" as opposed to variance in the case of PCA. The optimisation algorithm chooses each independent component so the order is more arbitrary than for PCA (where the order is defined by the magnitude of the eigenvalues). Often the ICA result is similar to PCA, but the independent components do not have to be orthogonal so can return different vectors to PCA. Figure 4.11b shows that the second iteration of the algorithm (details not shown) does not assign all the data to the second cluster (cyan). This is due to separation of the data that becomes more apparent when the data from the first iteration is removed. The result visualised in the original variables shows one ridge is well defined as in the PCA result, but only part of the other ridge has been identified (Figure 4.11c).

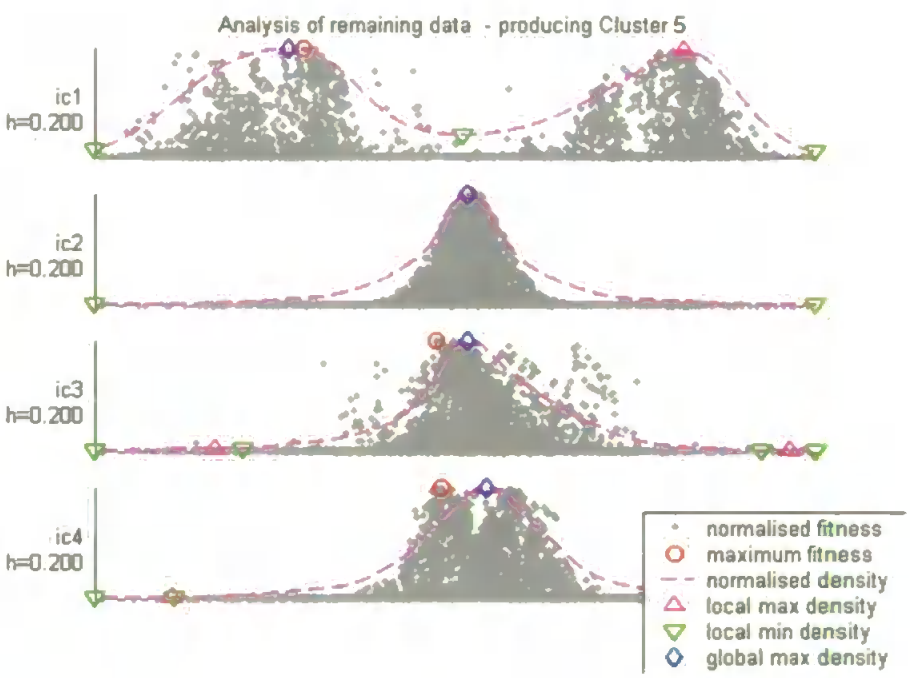


Figure 4.11a: KDE analysis of the independent components.

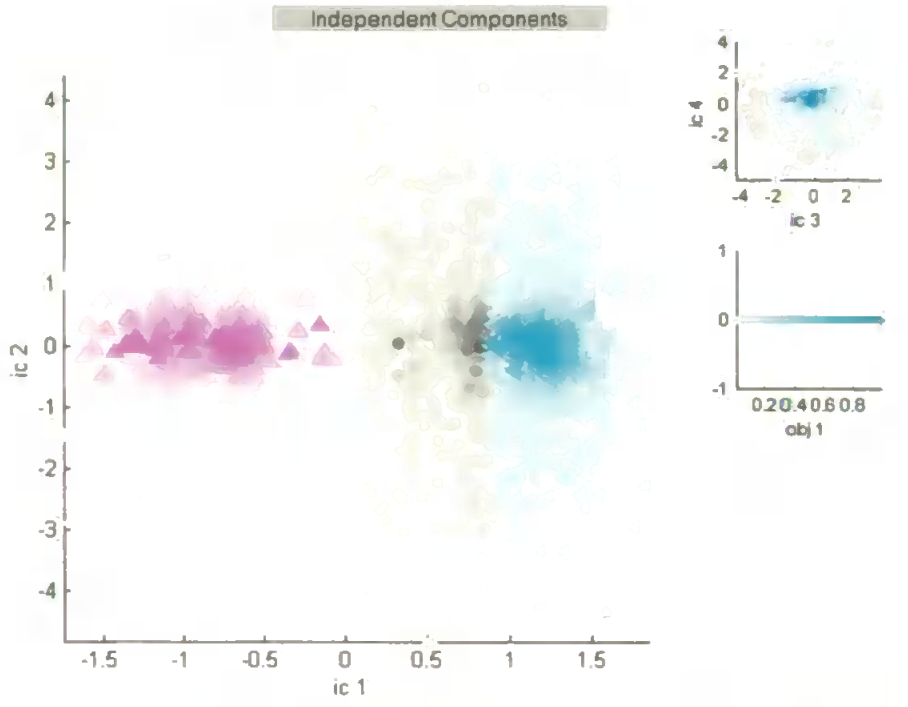


Figure 4.11b: Clusters identified in the independent component representation.

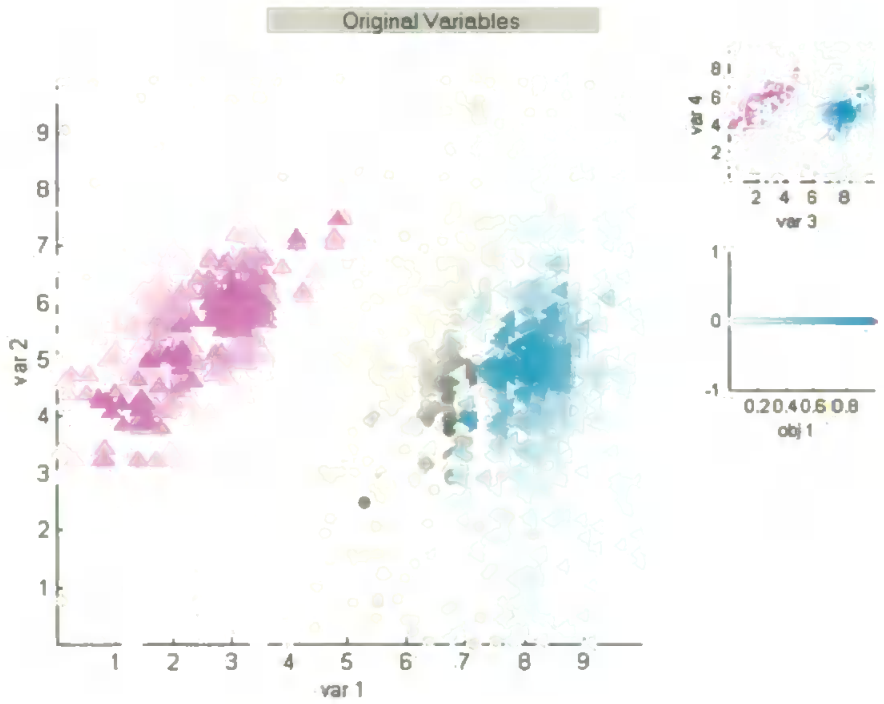


Figure 4.11c: Clusters identified using independent components, visualised in the original variables. Similar to principal component result, but not all the data has been classified.

The natural clusters are quickly identified by both PCA and ICA, usually in a single component. This can give a better partition of the whole data than the original variables that partition the data in a number of variables and often return parts of required clusters as perceived by a human. It should be noted that clustering using PCA and ICA on the Himmelblau data given in Section 4.3.2 returns almost identical results as the original variables, because the original variables are the natural vectors for the Himmelblau function (see Section 5.5). In general clustering in alternative coordinate systems can reveal different details of the data, although sometimes unexpected and maybe unwanted results will occur. However reasons for anomalies can be easily seen through the visualisation of the details; comparing result in the original variables is particularly instructive.

4.5 Conclusions

The univariate kernel density clustering algorithm can help to identify the main clusters in high dimensional data, the clusters are well defined in terms of the original or natural coordinate systems. In general as the number of dimensions increases the number of clusters identified will increase and the regions will become smaller. The GA converges on localised regions of the search space so the clusters produced by the GA are often spherical in nature even if the original fitness function is designed to be at an angle to the original data variables. This will be a problem for any clustering algorithm – the converging GA forms regions of high density that the KDE analysis is attracted towards. For some functions different GA runs will come up with very different data so it would be advisable to generate as much diverse information as possible.

Removing outliers from identified regions may also be beneficial, especially if they are of very low fitness or do not help to define the region very well. A lot of time could be invested in finding the right technique to cluster the data in terms of which outliers to

remove and which transformation vectors to use. More complicated criteria could be used to partition the data and to decide which views to show the user. However the motivation for the clustering algorithm was to provide the main clusters relating to engineering design data as quickly as possible, then allow the user to evaluate those clusters comparing fitness and robustness information. Rather than assuming the clustering algorithm is correct the user should be free to edit any clusters formed after analysis. The details inside clusters returned by the algorithm may be interesting or the user may wish to generate more data to fill in missing gaps.

A related question to ask is: when should the independent or principal components be used instead of the original coordinate system? The univariate KDE technique working on the original vectors often produces clusters that are too small and wrongly defined according to human perception. The natural coordinate systems usually produce large clusters that partition the whole data, but often misclassifies low fitness information, however it has been shown that if the low fitness information is removed the clusters do define the perceived regions well (Packham & Denham 2003). The independent components can provide different information from the principal components, although the result is less predictable. One of the goals of the clustering algorithm was to produce regions that were easy to define in terms of the original variables for engineering or manufacturing requirements. In alternative coordinate systems the definition of the region is related to the original coordinate system, but it remains to be seen whether this definition can be used to manufacture engineering design components. There may also be a problem with novice users understanding the new concept of clustering in alternative coordinate systems; presenting such information as default may make the user feel detached from the interactive process. However the superiority of the clustering using principal components (or less predictably the independent components) on some data suggests this technique should be retained in the overall system. Therefore it is proposed that the clustering

algorithm should work in the original variables by default and the use of PCA/ICA is left to the user.

It could be argued that the KDE-based technique described in this chapter is not a true clustering algorithm in the sense of one that naturally allocates all data due to the distance matrix between points, instead it uses coordinate systems that may or may not reveal the 'true' clusters in the data. It is acknowledged that traditional clustering algorithms could be adapted to take into account engineering type data in a similar way to that given in Section 4.3, but even the 'true' clustering algorithms will return different answers to a problem, as discussed in Section 3.5. Another criticism is the iterative nature of the algorithm, in particular removing data and recomputing the kernel density estimate at each iteration; clusters could be assigned immediately from the first iteration of the algorithm. Both of these criticisms are answered by arguing that the system is to be controlled by the user as opposed to generating many 'accurate' clusters and displaying them all or even choosing some to show. The philosophy of this system is to enable the user (especially novice users) to slowly assimilate the data presented in an understandable way related to the original variables. They can then decide whether to undertake further analysis of clusters already found, search for more clusters with the iterative algorithm or generate more data. The following chapter explains how the novel clustering procedure was successfully incorporated into an interactive design system for this purpose and subsequent chapters document how users responded to the system.

Chapter 5: Design of Interactive System

5.1 Introduction

Previous chapters have reviewed the requirements of an interactive evolutionary design system applied to engineering design and the human computer interaction techniques that are available to make such information available to the user. The main requirement of an engineering design system is to allow the free flow of information between the human and computer. The user should be provided with tools to guide the system about the search space, while the information coming out of the system should provide further understanding of the design space. The dual role of the human and computer means that the user can manipulate the information discovered and direct the search while at the same time the computer continues to generate new information based on the users' preferences by performing tedious tasks including quick statistical analysis.

This chapter will describe the tools chosen to support the dual requirement of the interactive system, summarised as follows:

1. **High dimensional visualisation**
2. **Flexible, understandable and easy to use interface**
3. **Fast exploration of the search space**
4. **Identifying clusters and searching for further data and clusters**
5. **Evaluating regions of high performance for robustness**

A system with these attributes will also support the kind of dual interaction proposed by Lund (2000, 2001) combining direct manipulation and interactive evolution. Lund (2001) concludes that direct manipulation by the user allows investigation in an intuitive and reliable way while interactive evolution is more likely to provide surprising solutions to the

problem (Lund used narrow interactive evolution as opposed to the broad interactive evolutionary system described here, however both methods can produce surprising and novel results). The tools implemented in the system are described in the order given above with reference to the literature. The actual interface, written in MATLAB ®, is presented at each stage with examples to show how the system is used in practice.

5.2 High Dimensional Visualisation

5.2.1 Choice of View Types

Engineering design problems are usually multidimensional, so it is important to be able to visualise all (or at least a subset) of variables and objectives so they can be seen in relation to each other. The visualisation techniques that do not distort the data and are most accessible to any user are variants of the scatterplot matrix (Chambers *et al.* 1983) and parallel coordinates (Inselberg & Dimsdale 1994a, 1994b) as concluded in chapter three of this thesis. A number of view options are available because it is often useful to see the same data from different points of view. In summary the available "view types" are:

1. Scatterplot Matrix
2. 2D Scatter (complete variable set in 2D plots with one enlarged to see details)
3. 2D+Fitness in Z-axis (as view 2 with the objective value shown in the z-axis)
4. 3D Scatter (three dimensional scatter plots with one enlarged to see details)
5. Parallel Coordinates

An example of the scatterplot matrix showing data from the four-dimensional Himmelblau function (see Section 4.3.2 and Figure 4.5) is shown in Figure 5.1. The coloured data points are clusters defined by the clustering algorithm described in Chapter 4. This view shows all the possible two-dimensional combinations, for example the first plot down in the first column is variable 1 against variable 2 (see enlarged version in

Figure 5.2), its mirror image is shown in the first plot on the first row: variable 2 against variable 1. Each variable is also plotted against the corresponding objective value (far right column and bottom row). It is a useful view, but details of each plot are difficult to see because of their small size and can take a long time to draw, this problem will be compounded as the number of variables increases.

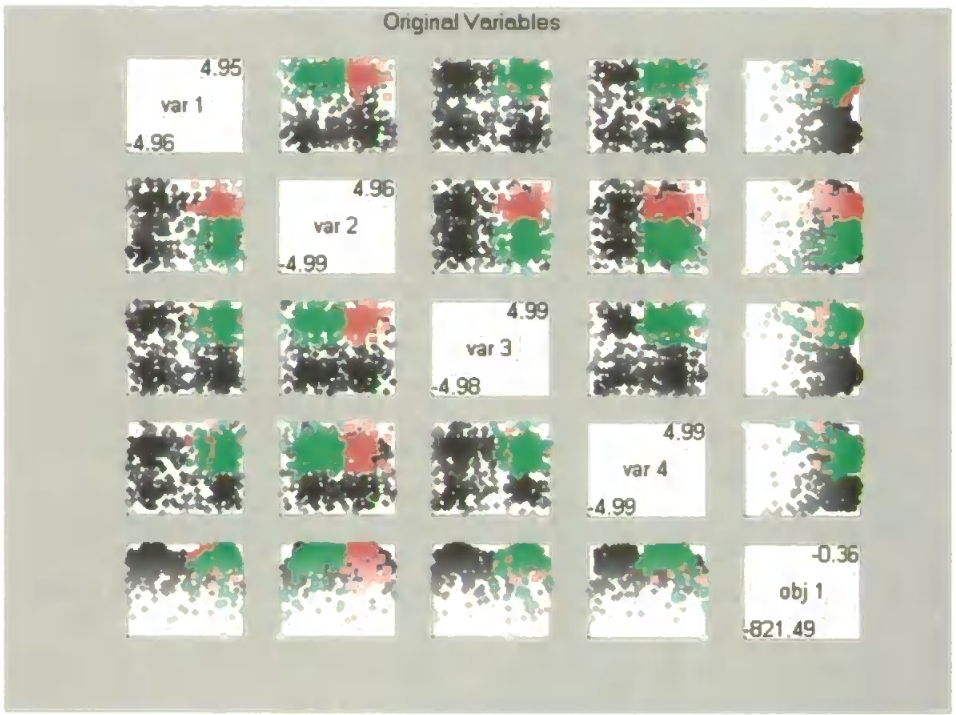


Figure 5.1: View 1 - the scatterplot matrix showing the four dimensional Himmelblau data. Variables ('var 1' to 'var 4') and objective ('obj 1') values shown.

The second view is similar to the scatterplot matrix but each variable is shown just once and two of the variables are shown in a larger plot so that details can be seen more clearly. In Figure 5.2 variables 1 and 2 are shown in the detailed plot, variables 3 and 4 on a smaller plot for reference. The objective values are shown in the second plot down on the right, the points drawn against a single value (arbitrarily zero) because there are no other parameters to show. It is possible to choose which variables are shown on the larger plot.

The third view (Figure 5.3) shows essentially the same information as the second view except that the corresponding fitness or objective value of each point is shown in the z-axis. This view is a substitute for surface diagrams that are often used in evolutionary

and engineering design literature; representing the fitness of each solution as a point in the third dimension was found to convey the same information whilst the colour of the points remains consistent with the other view types in the system.

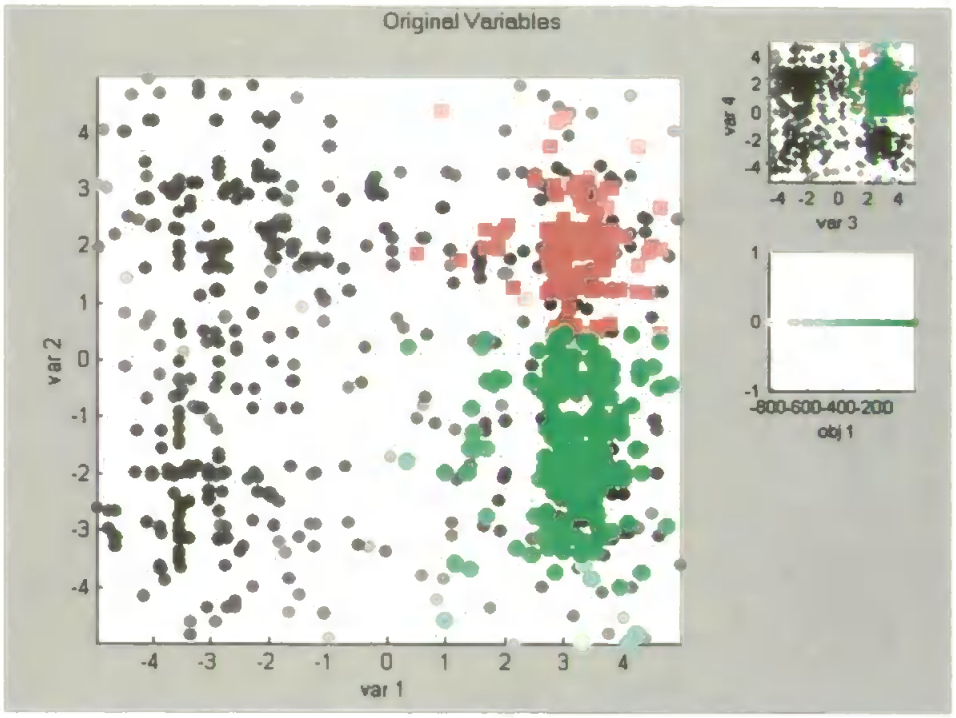


Figure 5.2: View 2 - 2D scatter diagrams with variables 1 and 2 in the enlarged plot, variables 3 and 4 top right and objective ('obj 1') alone. Fitness is also represented by the darkness of the data point.

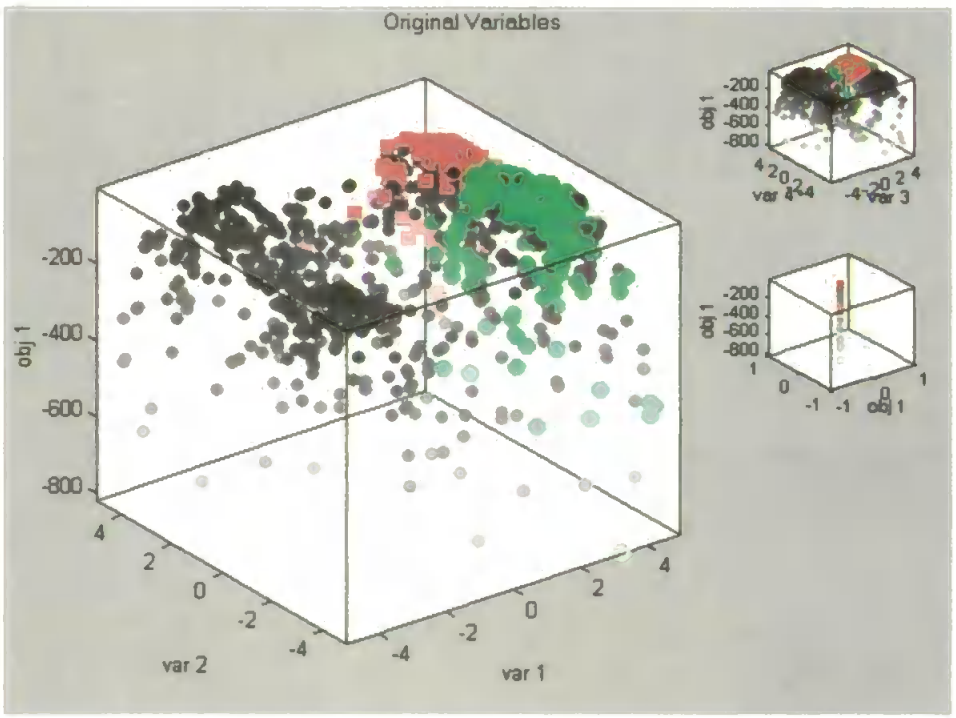


Figure 5.3: View 3 - 2D scatter diagrams + fitness in z-axis. The fitness or objective value ('obj 1') is shown on all plots.

View four is a set of three-dimensional scatter diagrams such that any combination of variables can be viewed in each plot (differing from view three where fitness is always shown in the z-axes). Again one of the plots is enlarged for detail (Figure 5.4). The three-dimensional views (3 and 4) are supplied because they are very accessible to humans who live in a three-dimensional world. However the exact position of a data point is uncertain when looking at the flat screen because it is not clear whether the point is at the 'front' or 'back' of the image, only the user's perceptual system interprets that information (rightly or wrongly) from all the data points. Because of this ambiguity the ability to select a region or number of data points with the mouse was not provided in the three-dimensional view types, but the user can select a region on any two-dimensional plot (view types 1, 2 or 5) and zoom in, then return to a three-dimensional view to help understand how solutions are related in variable and objective space.

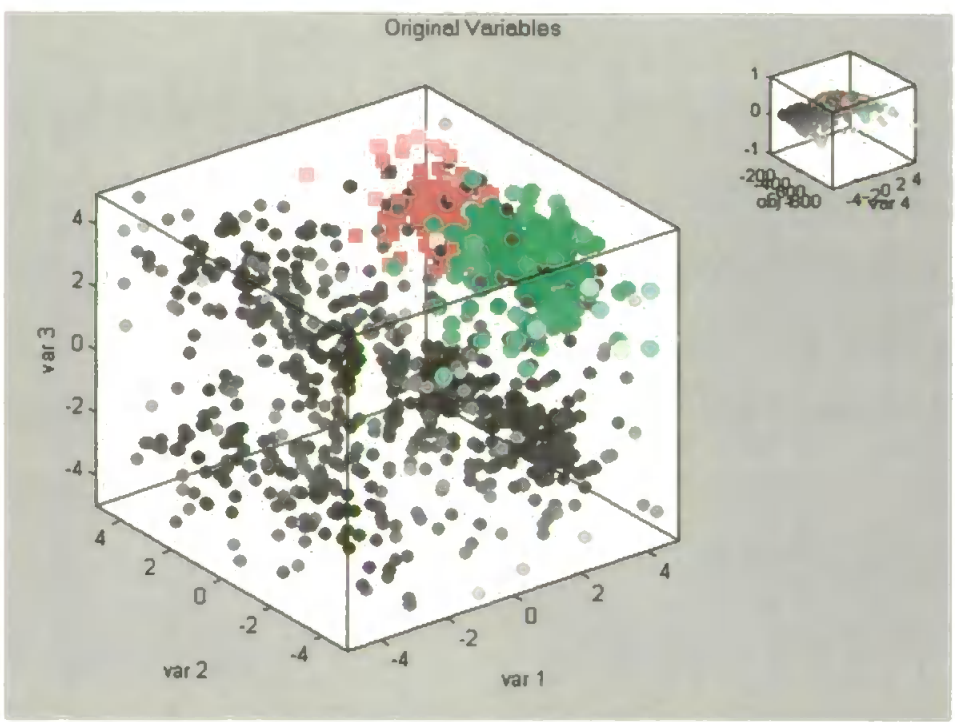


Figure 5.4: View 4 - 3D scatter diagrams, variables 1, 2 & 3 shown in detailed view.

The final view type is parallel coordinates (Figure 5.5, see also Section 3.3.5). All the variables and the objective can theoretically be viewed in one plot, the colour of each line is the same as the corresponding data point in the other views. The main problem with parallel coordinates is that the plot is still difficult to understand with a large number of variables and data, however in this system it is possible to change the order of variables, allowing different combinations to be compared, or choose subsets of variables or data lines to view as suggested by Swayne *et al.* (1998) and Siirtola (2000).

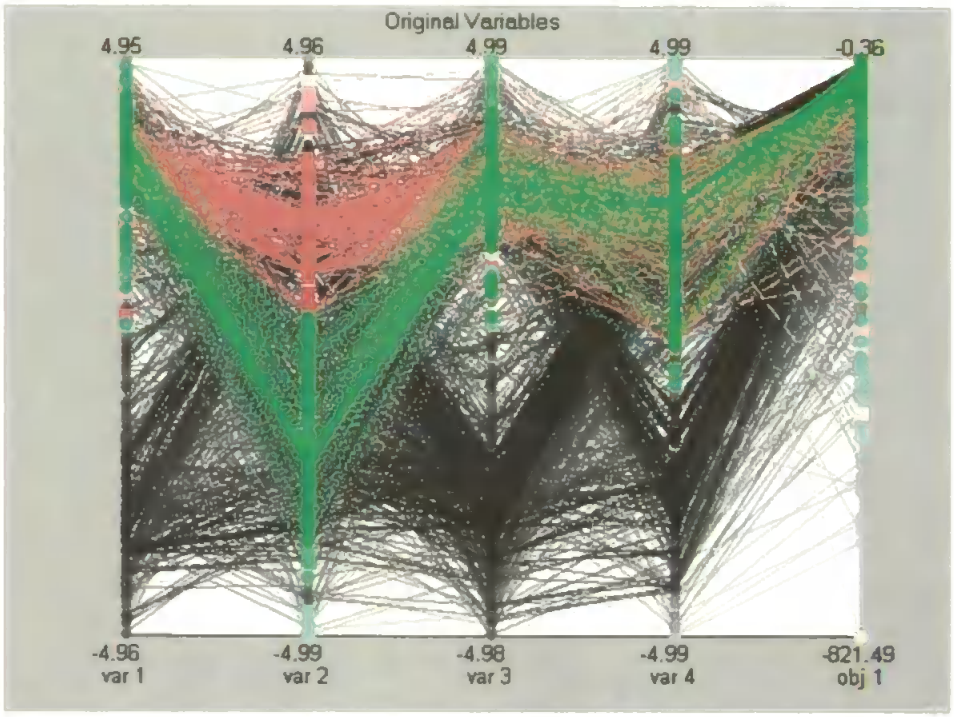


Figure 5.5: View 5 - Parallel Coordinates, each variable is given on a vertical axis, each is a solution, the colour of the lines corresponds to the data points in views 1 to 4.

5.2.2 Colour of Data

Colour is used to emphasise important clusters in the data and is also a good way to compare views of data, as the colours of each point will be the same in any view. In this way the colour coding is similar to brushing (Becker & Cleveland 1987, Section 3.3.2).

Every data point is initially drawn as a grey-black (colourless) dot or line to provide a neutral background, then a colour hue can be chosen for any new cluster identified by the system or user. Pure, or saturated, colours were not used because they can be overpowering when combined and cause eyestrain; so almost saturated versions of the spectral hues were used as base colours (red, green, blue, yellow, magenta, cyan etc.). Within each cluster the change in fitness is represented by change in brightness and saturation, the attributes of colour that most people can differentiate (Section 3.2.4 and Figure 3.2d). Saturation and brightness are varied simultaneously so that a light, unsaturated, colour is used for low fitness and a dark, almost pure, colour for high fitness, providing a uniform scale for any hue.

All the plots in this section use colour to indicate the clusters that were found by the system. The clusters that were identified by the clustering algorithm based on kernel density estimation (KDE), as described in Section 4.3.2. The complete data set ('All Data') generated by the GA is depicted as grey-black. The red cluster is the subset of 'All Data' that was identified by the KDE algorithm containing the solution of maximum fitness. The green cluster is the subset identified by the algorithm in the remaining data ('All Data' with red cluster removed) containing the solution of maximum fitness. More clusters could be identified in this way. The user is also free to create new coloured clusters or change the definition of clusters.

Clusters can overlap each other so each point may have a number of different colour hues associated with it. The display will also be affected by the order that colours are drawn on each plot as later colours will obscure any data drawn in the same place, however the order that colours are drawn can be changed (see Section 5.3.3 and Figure 5.11a). There are only a limited number of truly diverse colours that were chosen as default. For colour blind people the choice of red and green may not be suitable as the first

two colours, but it is also possible for the user to change the base colours. Relative difference in fitness, the most important attribute, should be apparent to most viewers.

5.2.3 Alternative Coordinate Systems

The variables of the data can also be viewed in natural coordinate systems such as the principal and independent components (Section 3.4 and Chapter 4). Figure 5.6 shows the principal components using the second view type. The coloured clusters have been mapped onto the principal components, but the objective value of each solution is unchanged allowing direct comparison of the views. The independent components can also be viewed in the same way (Figure 5.7 see Hyvärinen (2003) for FastICA code). Again any combination of these 'natural' variables can be viewed in each plot.

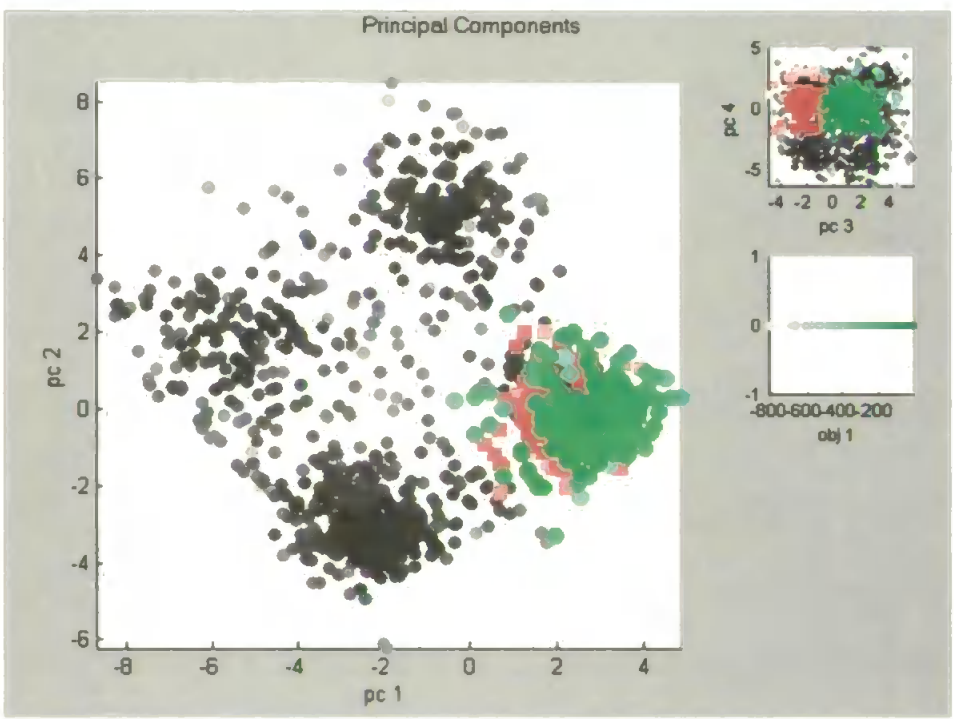


Figure 5.6: Principal components 1 to 4 ('pc 1' to 'pc 4') of data, coloured clusters correspond to the same data points given in original variables (Figures 5.1-5.5), objective value is unchanged.

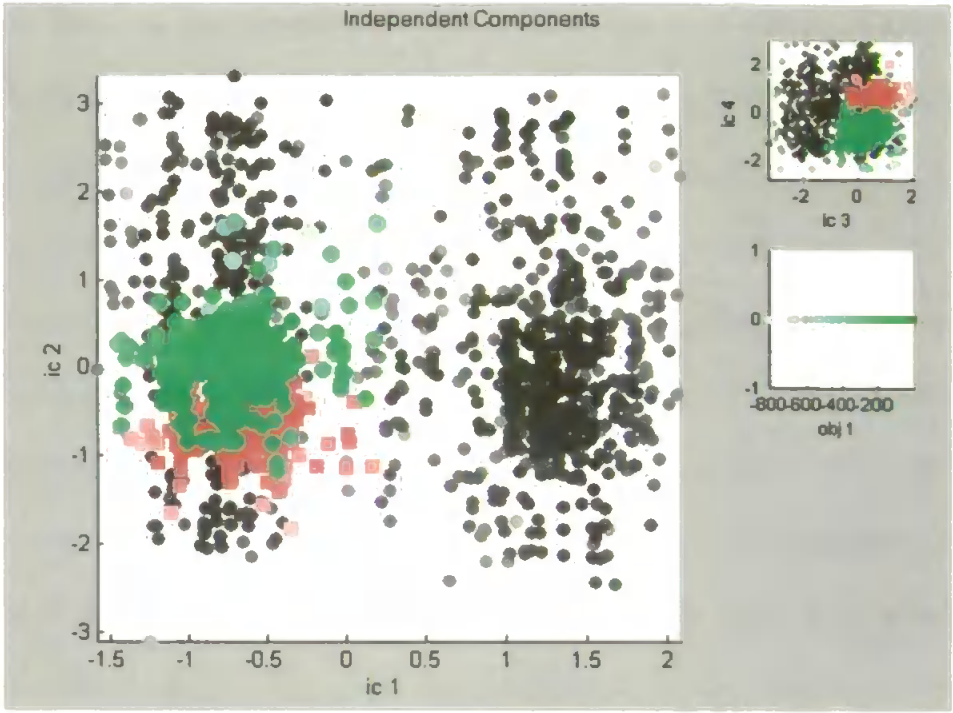
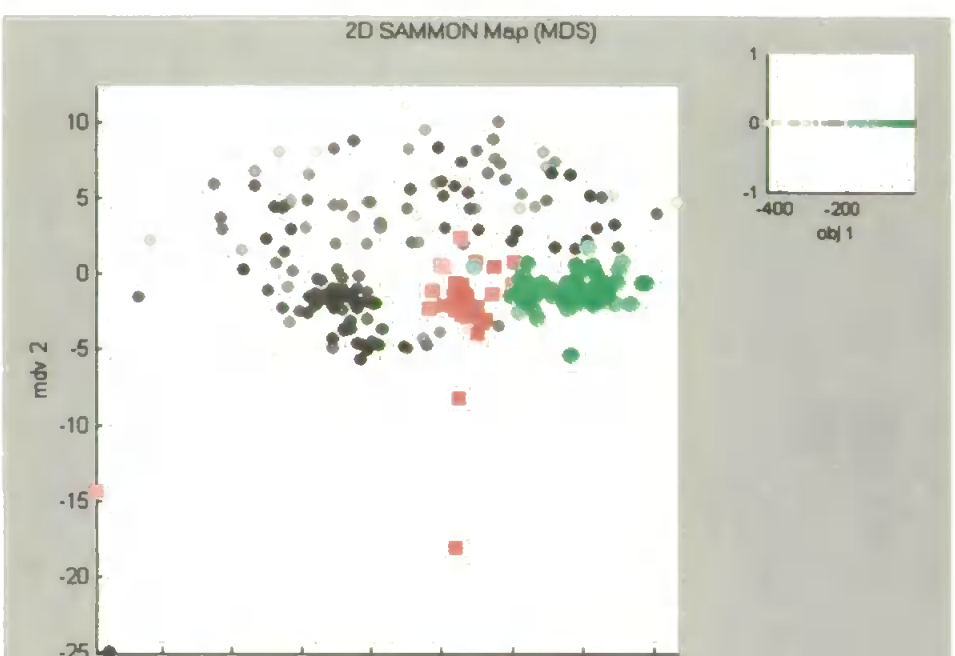
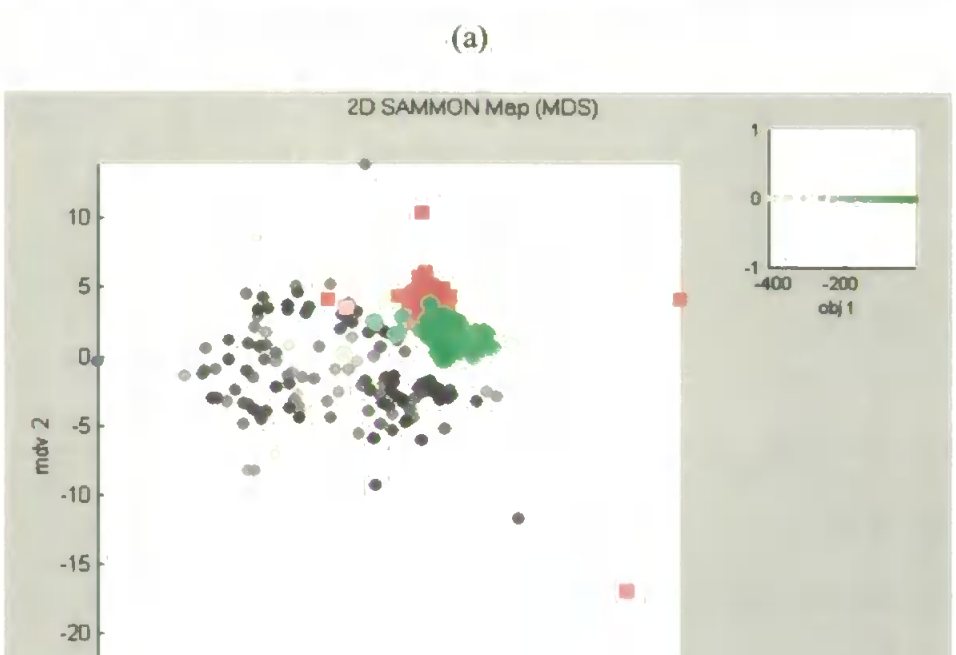


Figure 5.7: Independent components ('ic 1' to 'ic 4'), of the Himmelblau data, objective ('obj 1') unchanged. Computed using the FastICA algorithm (Hyvärinen 2003) supplied under the GNU licence (GNU 2003).

A two dimensional representation of the clustering is also available to see how clusters relate to each other in terms of Euclidean distance. The dimensionality reduction technique SAMMON mapping (Sammon 1969) is used to generate a two-dimensional output optimised such that the distance between points in the output are as close as possible to the original distance matrix. The actual distance between points can be assessed using this technique. The drawback of multidimensional scaling (MDS) techniques is the time taken to compute the distance matrix, so in this system a subset of data points is used to find the mapping. The result of SAMMON mapping is also unpredictable because it is an optimisation routine; given an initial data set, a number of results could be found using the same parameters. The routine terminates when the error between the true Euclidean distance and the output is below a certain threshold, but the error for individual data points may be quite high.



(a)



(b)

Figure 5.8: Two runs of the SAMMON mapping routine using the same initial data. A subset of 250 Points is used to reduce computation cost. Two multidimensionally scaled variables ('mdv 1' and 'mdv 2') and the objective value of each point is shown. Computed using code supplied to the SOM toolbox (Vesanto *et al.* 2000) published under the GNU (2003) licence.

Figure 5.8 shows two results of running the SAMMON mapping routine on a subset of the Himmelblau data (see the SOM Toolbox (Vesanto *et al.* 2000) for code). Note that the two results are different although the structure of the data is retained in both. In each figure, the two multidimensionally scaled variables ('mdv 1' and 'mdv 2') are shown in the main plot, with the corresponding fitness of each point given in the other plot ('obj 1'). The data could be visualised using any of the view types mentioned earlier. Again users can select clusters seen on the SAMMON map display and see the corresponding solutions in the original variables.

5.2.4 Summary

To summarise this section, the system has successfully employed a number of traditional techniques for the difficult task of high dimensional visualisation. Two and three dimensional scatter plots and parallel coordinates can be manipulated as the user wishes. The hue of colour is used to highlight important clusters in the data and variation in brightness and saturation of colour is used to indicate the relative fitness of solutions. The data can also be viewed from alternative coordinate systems such as the principal or independent components or a 2D multidimensionally scaled SAMMON map. The alternative coordinate systems can show clustering that the original variables do not reveal; colour coding and comparing different views or coordinate systems provides a powerful way to learn more about the data.

5.3 Flexible, Understandable and Easy to Use Interface

5.3.1 Introduction

For any novice users learning a new interactive visualisation system the interface should be as simple as possible and self-explanatory (Shneiderman 1998, p. 68). However for the system to be truly interactive and to allow the user to manipulate the data as much as possible, a large amount of functionality is needed in the system. The design of the

interface was inspired by the mantra: *Overview, Zoom and Details on Demand* (paraphrasing Shneiderman 1998, pp. 523) and evolved as new ideas and algorithms were added (in a similar way to how the GGOBI system (Swayne *et al.* 2001) was developed).

The system was aimed at intelligent novice users and knowledgeable users, so common tasks are available as buttons whilst further detailed commands are found in window menus and dedicated dialog boxes. Ideally the interface would be designed by the intended users using the participative approach (Mumford 2003). However an interactive system combining evolutionary computing and engineering design has never been designed before, so there were no users to consult. Therefore the interface evolved following the visualisation guidelines given in Chapter 3 and modified as problems were encountered and solved. The interface was evaluated by novice users and engineers working on their own design problems as described in Chapters 6-8.

5.3.2 Navigator, Overview and Moreview Windows

The main controller of the system is called the Navigator Window whilst the overall view or map of the search space is called the Overview Window. Figure 5.9 shows these windows after the user has chosen to start the system on the four dimensional Himmelblau function, after 20 generations of the genetic algorithm (GA) the Overview Window displays the familiar result in the default view (view type 2). The user can select a region of any two dimensional view on the Overview Window and choose a command from the Navigator such as 'Run GA', 'Find Clusters', 'Zoom In' or 'Delete' (the data). Alternatively the user could choose to view the data using a different view type or coordinate system by selecting from the menus on the Overview Window. The AxesOrder menu brings up the dialog (right of the Overview Window in Figure 5.9) that allows the user to choose the order of variables (objectives) or deselect specific variables (objectives) to display.

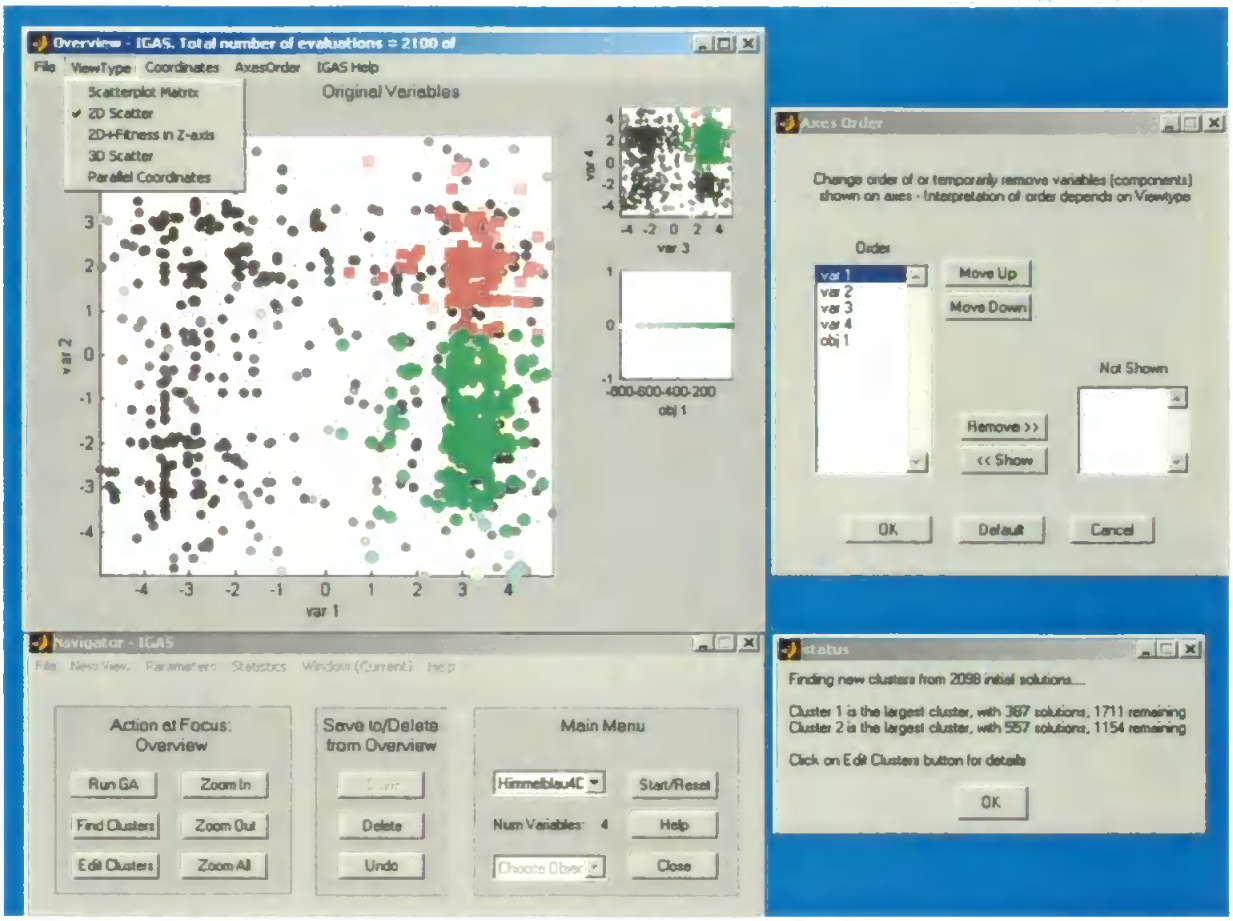


Figure 5.9: Navigator Window and Overview Window. Order of variables can be changed or variables temporarily removed using the 'Axes Order' dialog.

New windows are necessary either to view new data generated by further GA runs or to view data from different points of views. In Figure 5.10 the user has chosen a region of data with the mouse in the Overview window and opened a new window using the New View menu on the Navigator. The new view is called Moreview2 and shows the data using the '2D+Fitness in Z-axis' view type. Alternative coordinate systems can also be chosen from the New View menu. Individual details of specific solutions can be displayed by pressing the right mouse button. Details of individual solutions, statistics of clusters and a summary of all the data can be accessed and are displayed in the MATLAB command window. Other menus on the Navigator Window allow the user to change the parameters of algorithms and select which window is to be the Current View. Hence the *Overview, Zoom and Details on Demand* concept is strongly followed with this interface.

In order to prevent errors dialog boxes are used to confirm important commands (such as save, delete, run GA at new limits). Backtracking with undo and redo commands would be desirable but this requires a lot of memory so these are only incorporated into the important save and delete commands. Status messages tell the user what actions the system is performing when busy and keep the user informed of what the algorithms have discovered (see box next to Navigator in Figure 5.10). Help files are available on all windows and context specific help is available when the mouse is held over a button or part of a dialog box.

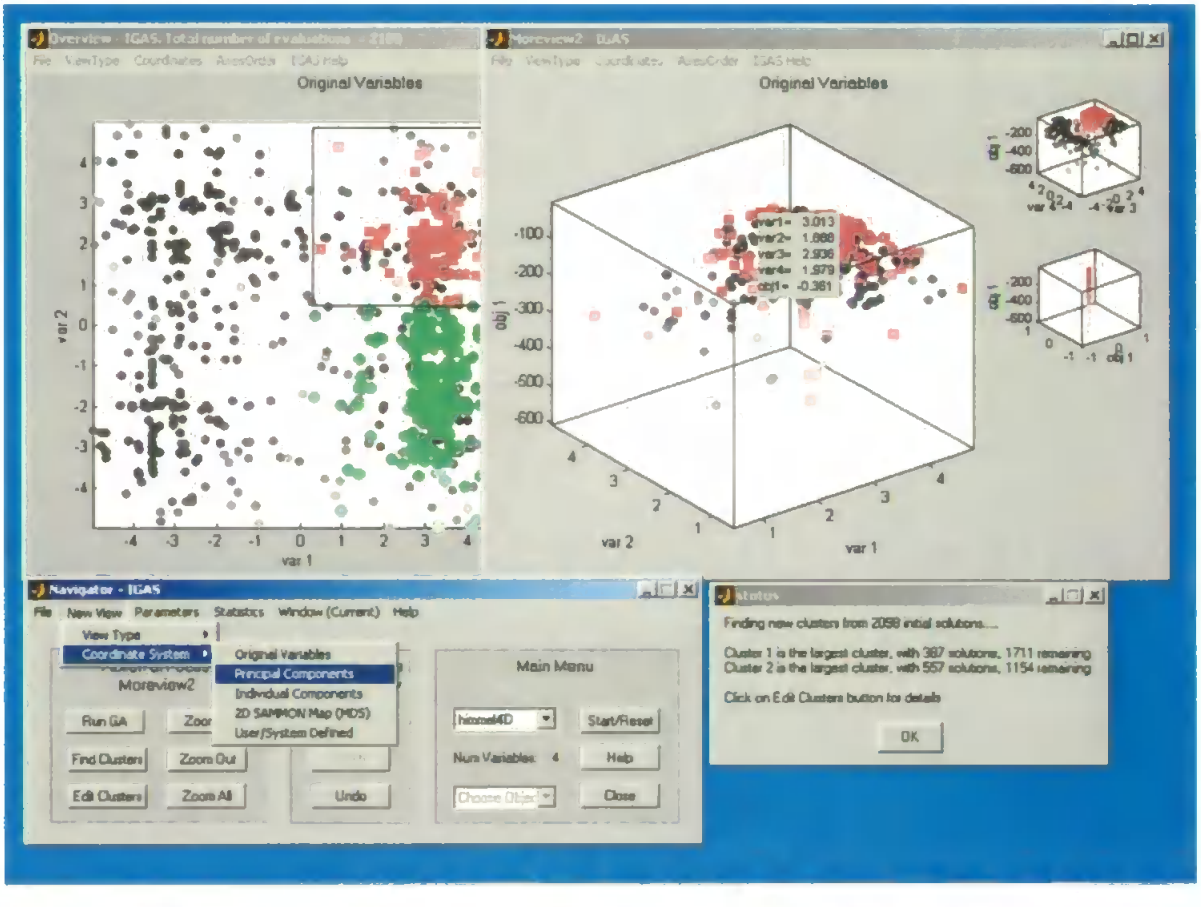
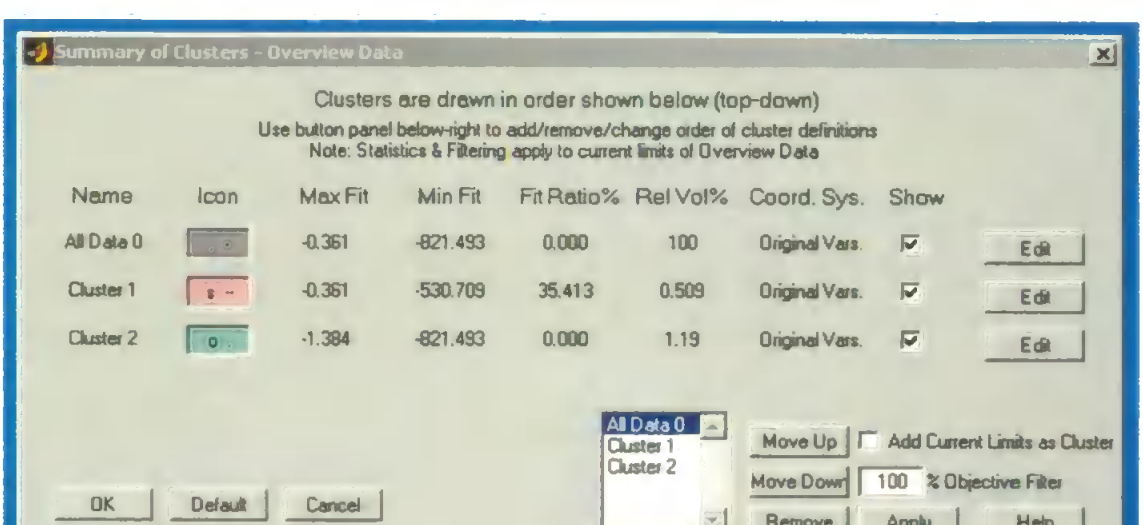


Figure 5.10: New views supported to provide Zoom and Details on Demand. Region selected by the mouse in the Overview, zoomed in version displayed in Moreview2, note the full range of variables 3 and 4 remain, but there is no green data. Details of a data item can be viewed by clicking the right mouse button over a point.

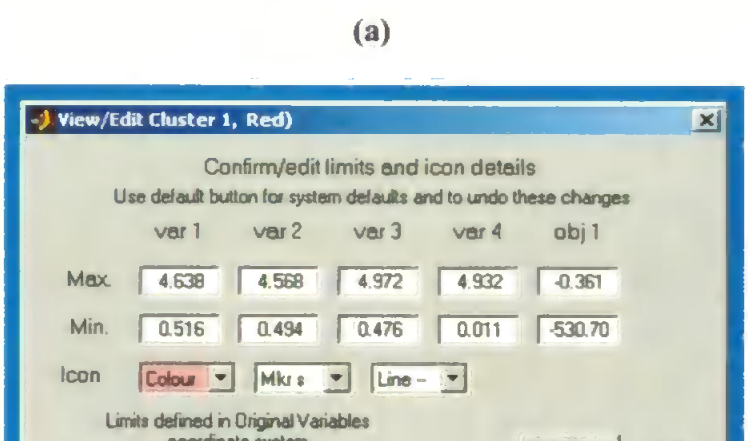
The Save/Delete panel on the Navigator (see Figure 5.9) controls how information is added and removed from the Overview Window. The user can select a number of solutions in the Overview Window using the mouse and choose to delete them permanently, alternatively the user can choose all or a subset of the data in a Moreview Window and add the information to the Overview Window. An 'Undo' button is available if the user has mistakenly added or deleted data. By default all the information generated by each GA is kept for the user to consider and analyse, but after a while the user may decide certain information is not needed or helpful (particularly low fitness information). It is also possible to save data to file using the traditional File menu on the Navigator and to load previously saved data.

5.3.3 Dialogs to View or Edit Clusters, Relevant Statistics Included

As mentioned in Section 5.2.2, the colour coding of clusters is similar to brushing; the use of colour is fully exploited in this system by allowing the user to change the definition of clusters or add new ones manually. The 'Edit Clusters' button on the Navigator Window brings up the Summary of Clusters dialog box (Figure 5.11a) that shows a summary of all clusters including their colour, icon and general statistics. The user can choose which clusters definitions to 'Show', when not checked the colour will be temporarily removed from the display. Cluster definitions can be added and removed from the list (see button panel bottom right of Figure 11a). The clusters are drawn in the order shown in the Summary Window, so some colours may be obscured by those drawn later on if cluster definitions overlap; so the drawing order of the clusters can also be changed. The 'Objective Filter' checkbox is used to automatically change the definition of clusters; the definition of clusters (and whether to 'show' them) is crucial to the functionality of the system as the GA and clustering technique can be controlled using these definitions, as explained further in Section 5.5 and 5.6.



(a)



(b)

Figure 5.11: (a) Summary of Clusters dialog and (b)View/Edit Clusters dialog. Important statistics are shown in (a), the user can choose to 'Show', change the drawing order, filter or permanently delete clusters or add a new one. The 'Edit' button brings up the individual cluster information (b) allowing changes to the variable limits, colour and shape of the icon, and access to more detailed statistics.

The statistics shown in Figure 5.11a are intended to help the user make informed decisions about the quality and potential robustness of each cluster. The maximum and minimum fitness inside each cluster are given and the fitness ratio ('Fit Ratio%') between them is computed with reference to the overall data, given as a percentage:

$$f_R = \frac{\min(f) - \min(F)}{\max(f) - \min(F)} * 100 \qquad \textit{Equation 5.1}$$

where f_R = fitness ratio of a cluster

f = objective values of solutions in a cluster

F = objective values of all the data in the figure

The relative hypervolume ('Rel Vol%') in variable space of each cluster is also given, again with reference to the overall data and as a percentage:

$$V_R = \frac{\prod_{i=1}^{v} [\max(x_i) - \min(x_i)]}{\prod_{i=1}^{v} [\max(X_i) - \min(X_i)]} * 100 \qquad \textit{Equation 5.2}$$

where V_R = relative volume of the region defined by a cluster

v = number of variables/components

x = variable/component values of solutions in a cluster

X = variable/component values of overall data in the figure

The values of x and X could be given in the original variables or some alternative coordinate system ('Coord Sys'), found using principal or independent component analysis (PCA/ICA), for example. It could be argued that the generalised variance (Section 3.4.3) of the cluster should be used to estimate the volume, but this value depends on the number of data points and assumes the data is normalised so is not comparable between different sets of data. Instead the relative volume of the 'region' encompassed by the cluster allows comparison between different data sets that may be generated for the same problem.

The relative volume of the region tells the user how large it is, while the fitness ratio informs how much the fitness changes inside the cluster compared to the overall data. By comparing relative volume and fitness ratio the user can estimate the likely robustness of the region, but should be aware that more solutions may be needed to confirm the

estimate. Maximum fitness and 'robustness' of each region can then be compared and opinions about the relative merits of each region formed.

Details of individual clusters can be viewed and edited by pressing the 'Edit' button in the Summary bringing up the View/Edit Clusters dialog in which the limits, colour and type of icon can be checked and changed. Colour deficient viewers could choose any suitable colours using this dialog box and different shapes of data points can also help to differentiate between clusters ('Mkr' Figure 5.11b) as well as alternative 'Line' types for the parallel coordinate display. Limits of the cluster can be edited using this dialog (even if the cluster is defined in another coordinate system such as the principal components). Details and statistics of the cluster can also be displayed in the MATLAB Command Window.

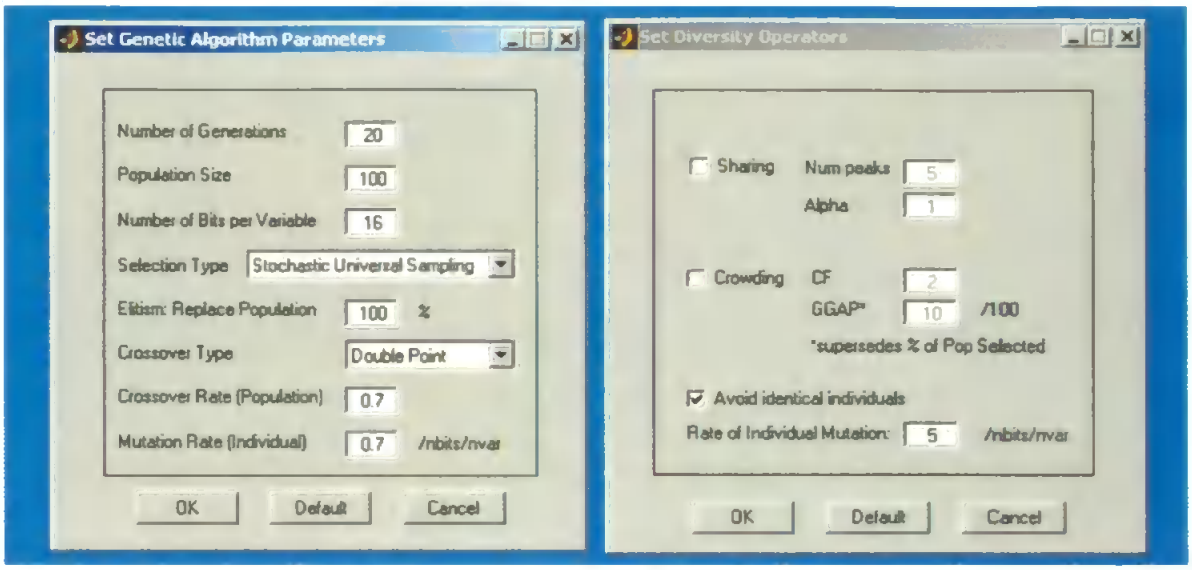
5.3.4 Summary

An easy to use interface has been designed that includes a Navigator Window, an Overview Window that keeps all data saved by the user and Moreview Windows that contain data from new searches requested by the user or are clones of other windows. The data can be saved to file and warning messages will prevent the accidental loss of data. Clusters of solutions are identified by different colours and icon shapes and represented consistently in each cloned window. The user is free to change the definition, colour and icon of clusters. A summary of the clusters shows statistics that help the user make informed design decisions and allows details and limits of the clusters to be changed. This design is a combination of many well-known techniques put together to help engineers understand the information found. The rest of the chapter explains how the GA is used to generate diverse data and how the system can be used to exploit that information to further understand the engineering design problem.

5.4 Fast Exploration of the Search Space

For many engineering design problems the calculation of the objective function will be very time consuming and the number of calls to this function should be minimised. A genetic algorithm (GA) that is run for a low number of generations in specific regions chosen by the user is a fast, powerful exploratory tool. This method exploits the GA's power of quick convergence as Louis & Tang (1999) did by breaking the travelling salesperson problem up into a number of sub-problems. Another advantage for quick GA runs is that they are not so dependent on getting the parameter values right first time (Ross & Corne 1994). Conversely this approach removes the full evolutionary theory from the system, the GA becomes a tool rather than an ongoing algorithm that will eventually find new solutions. Nevertheless the whole system will allow extended exploration of the search space because of the combination of the user with the GA. The GA code used in the system was modified from the GA Toolbox for Matlab (Chipperfield *et al.* 2002), published under the GNU General Public Licence (GNU 2003).

As mentioned in the previous chapter it is important to keep the diversity of the GA high to improve exploration of the search space. Sharing (Deb & Goldberg 1989) and Crowding (De Jong 1975) as described in Section 2.3.3 are the classical techniques designed to improve diversity, but both need careful tuning of the parameters (Miller & Shaw 1996). A person with knowledge of GAs may want to adjust the GA parameters and diversity operators and these have been made available (Figure 5.12), but this activity is time consuming especially if the fitness function is very expensive. Most engineers have little knowledge of GAs, but would like to see many diverse solutions to their problem generated as quickly as possible without duplication, although good regions of the search space should also be exploited and sampled as much as possible. So the parameters of the GA were chosen carefully to allow as much diversity as possible without affecting speed whilst exploiting the good regions found.



(a) **(b)**

Figure 5.12: GA Parameters (a) and Diversity Operators (b).

The default parameters shown in Figure 5.12a could be described as those found in a 'simple' GA. The crossover and mutation rates are those suggested in the GA toolbox for binary representation (Chipperfield *et al.* 2002, pp. 2-15, 2-46). Mutation rate is given as the expected number of mutations per individual, which can be converted to the probability of mutating a bit by dividing the rate by chromosome length - that is the number of bits per variable (nbits) multiplied by the number of variables (nvar). Most researchers suggest the probability of mutation should be in the range 0.001 to 0.01 per bit (Ochoa *et al.* 1999, Bäck 1993, Hesser & Männer 1990), so assuming chromosome length is between 70 and 700 the given mutation rate of 0.7 is reasonable. The selection technique used is stochastic universal sampling (Baker 1987) because it reduces stochastic error and selects individuals closer to the true rate deserved by fitness than most other selection routines (Pohlheim 1996), therefore producing a more accurate interpretation of the fitness landscape.

The main departure from the simple GA is the introduction of the scheme to mutate chromosomes that are identical to those already found (as described in Section 4.3.1), ensuring that no duplicate solutions are generated during the GA run. Checks for

duplication are performed at the chromosome level. The level of mutation on duplicated solutions will have a significant effect on the results of the GA. A low mutation rate makes solutions stay near to the original solution, probably moving along peaks, while a high mutation rate would result in almost random search. A 'moderately high' mutation rate has been chosen as a trade-off, increasing the probability of discovering new high performance regions whilst assuring that the search space near to the original solution will be further sampled. The default mutation rate in duplicated individuals is 5 (the expected number of mutated bits); that is the probability of mutation per bit is 5 divided by chromosome length (Figure 5.12b). This is moderately high assuming the length of the chromosome is between 70 and 700. For very long or short chromosomes this mutation rate should be adjusted; in particular the rate is impossibly high if the chromosome length is less than 5.

In summary fast exploration of the search space is achieved by running a GA for a low number of generations with default parameters and mutation of duplicate solutions. Dialog boxes are available to allow changes to the GA parameters and include diversity operators such as crowding and sharing for GA interested users. A mutation scheme is included in the default parameters that retains the exploitative power of the GA, returning a diverse set of solutions to the problem whilst ensuring a number of solutions are likely to be generated in high performing regions. This system should give engineers a number of design options with enough information to evaluate those options in terms of robustness. The user may wish to further analyse a region or explicitly avoid a region of the search space that has already been found, this requirement is discussed in the next section.

5.5 Identifying Clusters and Searching for Further Data and Clusters

In an interactive environment the obvious clusters should be presented to the user without undue delay; refinement of clustering or classification of ambiguous points can be performed by the human or computer at a later time (as suggested by Levinson *et al* (1979)

in the ISODATA environment). It is not viable to show all the different possible projections or cluster combinations; ideally a user would have the option to demand specific techniques to cluster the data, but this requires familiarity with clustering techniques and domain knowledge. Discovering and identifying robust regions was covered in Chapter 4, values for the relative volume and fitness ratio will help the user decide which clusters are worth investigating. Clusters that have low hypervolume but relatively high fitness are also worth looking into because they could turn out to be robust regions that the GA has not sampled properly or of very high fitness.

By default the kernel density estimation (KDE) based clustering analysis is performed on the original variables because they will be most familiar to the user and easy to understand; analysis on the principal and independent components may reveal more natural clusters, but could confuse the novice user so is available on demand as explained in the conclusions to Chapter 4. Default parameters for the clustering technique will be supplied and the basic parameters can be changed by the knowledgeable user allowing alternative results (Figure 5.13), the user can also choose to see exactly how the clustering algorithm came to its conclusion by checking the 'Show clustering details' check box.

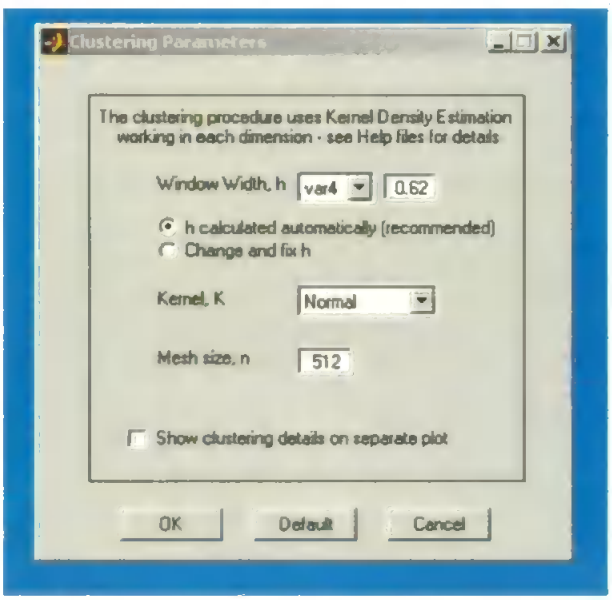
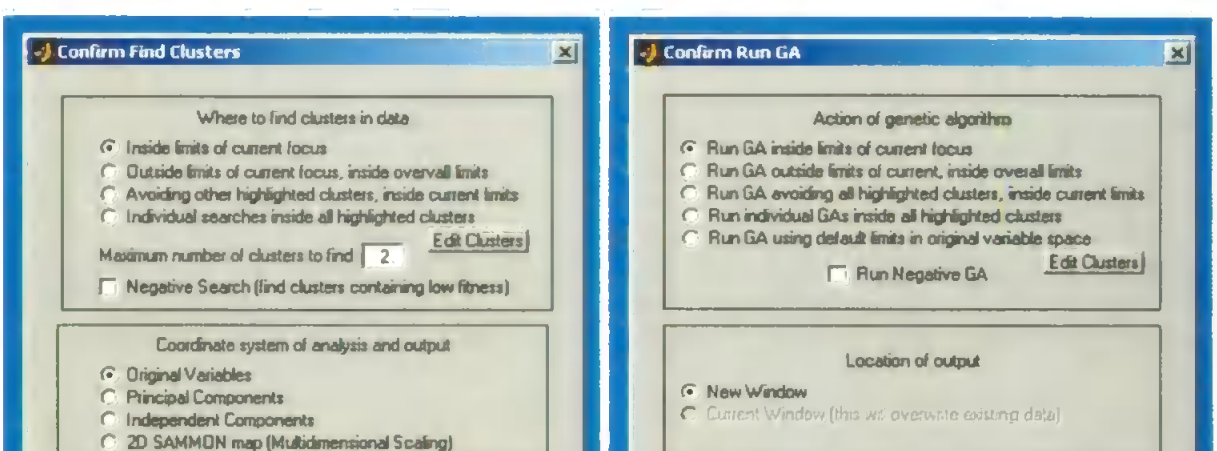


Figure 5.13: Basic Kernel Density Estimation Parameters for Clustering (see Chapter 4 for details).

After an initial GA run and the first KDE analysis there are a number of possible actions the user could take. The user could change the definition of clusters (using the 'Edit Clusters' button and dialog, Figure 5.11b) found by the original KDE analysis or choose a region of the search space with the mouse and perform a further clustering analysis. Figure 5.14a shows the dialog that appears when the 'Find Clusters' button is pressed on the Navigator window. The user can choose whether to find clusters 'inside' limits of the current focus, 'outside' those limits, 'avoiding' all highlighted clusters already discovered or search for more clusters inside each 'individual' highlighted cluster already found. 'Highlighted' clusters are those whose 'show' option is checked (Figure 5.11a) allowing the user to choose regions of the data to work on without permanently deleting cluster definitions. The maximum number of clusters to be generated is also under the user's control, unless the clustering algorithm runs out of data before that number is reached. The user can also choose which coordinate system to perform the clustering in (bottom of dialog, Figure 5.14a).



(a) (b)

Figure 5.14: Dialog boxes to give user control of future events. Options to choose: (a) where to find new clusters with KDE analysis, (b) where to search with GA.

Similarly the user could decide to perform another GA run either inside or outside limits defined by the user or inside/avoiding clusters already defined (Figure 5.14b). The

clusters have been defined by the clustering technique either in terms of the original limits or the transformation matrix of an alternative coordinate system and corresponding boundaries. The GA and Clustering dialog boxes (Figure 5.14) also contain buttons to 'Edit Clusters' and the opportunity to change algorithm parameters. The 'Run Negative GA' and 'Negative Search' are innovative additions to the system that help to evaluate the robustness of regions of the search space, explained further in Section 5.6.

Figure 5.15 shows the clustering analysis performed on the principal components of the 4D Himmelblau data using the option to 'avoid' other highlighted clusters. The option to show clustering details was checked from the 'Parameters' menu and the additional plot shows how partitioning the data along each principal component formed the blue cluster. The blue and yellow clusters shown here are different to those shown in Figure 4.7 because these are found using the principal components. As described in the previous chapter, clusters are defined by analysing the densities along each variable and the cluster containing the highest fitness is identified, assigned a colour (blue) and temporarily removed from the data so the next cluster can be found (yellow). Note that despite the difference in KDE analysis these clusters are very similar to those found in the original variables (compare left-hand window in Figure 5.16 with Figure 4.7g).

If the user chooses to run the GA whilst excluding certain clusters, the solutions that fall into that cluster can be given a reduced fitness or death penalty to stop them being selected in the next generation (in a similar way to the sequential niche technique (Beasley *et al.* 1993)). If the search space is sparse the GA may have trouble finding other peaks and will probably be drawn towards the outskirts of the original peak (as Mahfoud (1995) described in his criticism of Beasley *et al.* (1993)). Therefore it may be necessary to expand the cluster that the GA should ignore, currently this is left to the user to assess and change manually but some automatic technique could be devised.

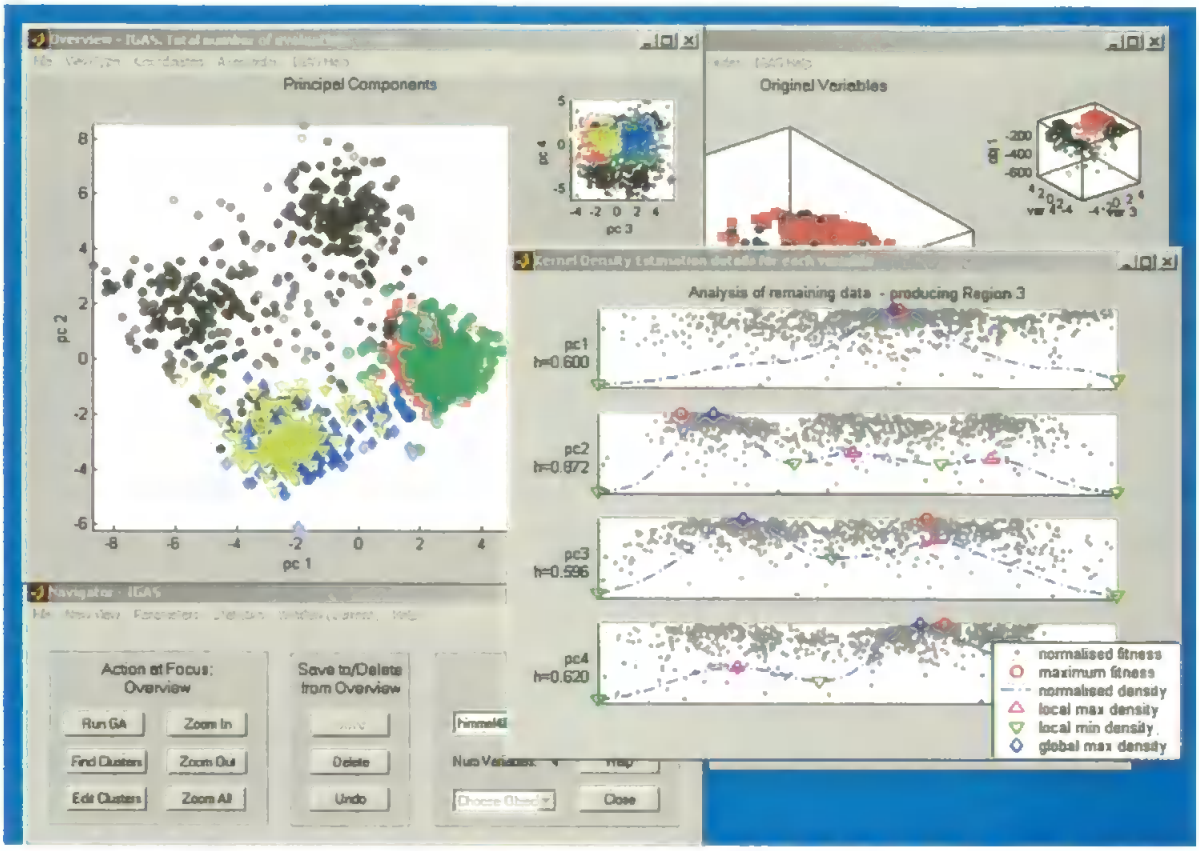


Figure 5.15: Further clustering performed on the 4D Himmelblau data. Options used: 'Avoiding other highlighted clusters', 'Principal Components' (Figure 5.14a) and 'Show clustering details' (Figure 5.13).

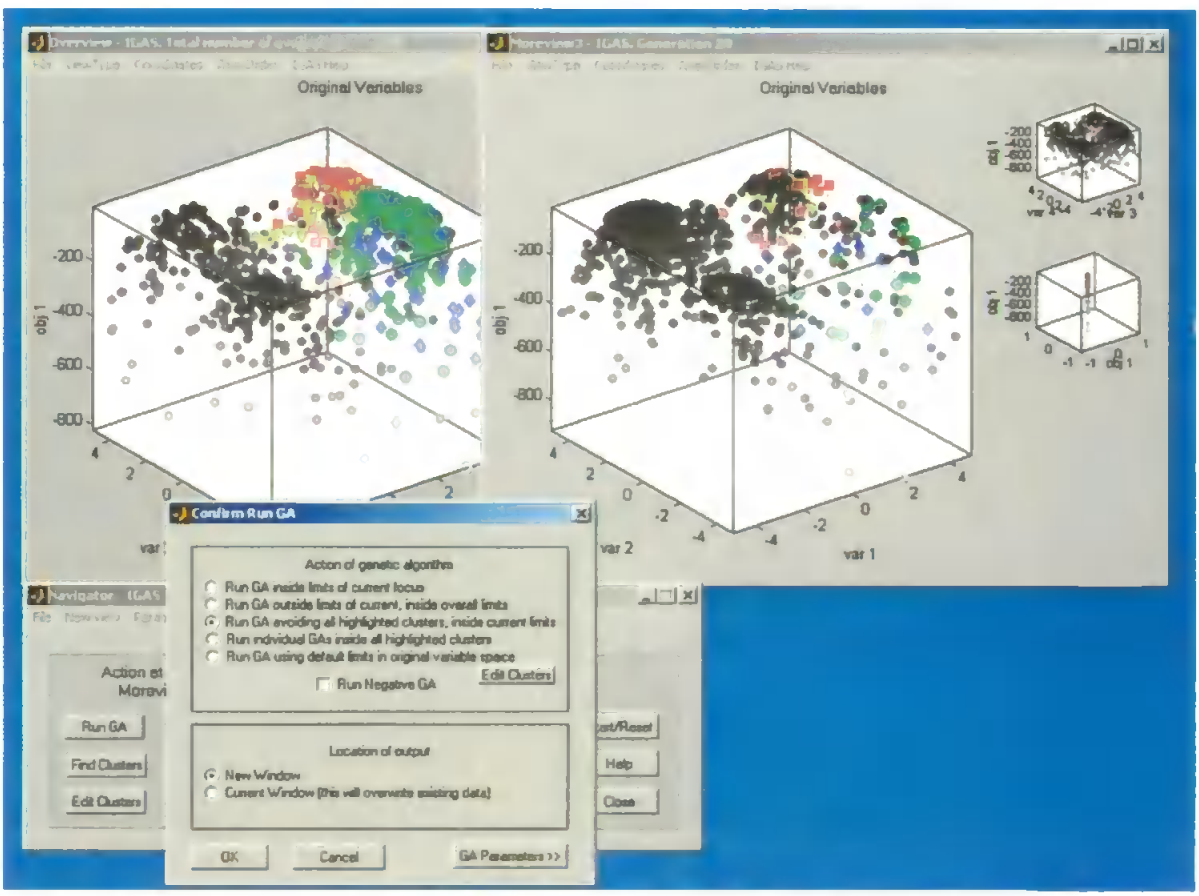


Figure 5.16: GA avoiding highlighted clusters (output in Moreview3), more grey-black data points found away from the coloured regions.

Figure 5.16 shows the output of a GA running with the option to avoid the four regions defined by the clusters found previously. The regions defined by the principal components are mapped back to the original coordinates using the transformation matrix and limits whilst the GA is running. Moreview3 in Figure 5.16 shows how solutions still appear in those avoided regions because they were evaluated during early generations of the GA before being given the death penalty, in later generations solutions outside the avoided regions are more likely to survive.

5.6 Evaluating Regions of High Performance for Robustness

The visual display of clusters and the available statistics give an indication of the relative robustness of those regions defined by the extent of the clusters. However for most problems the GA will have only sampled a proportion of the available solutions in each region – for a problem with continuous attributes there will be an infinite number of possibilities. A region may contain some very bad solutions that the GA has not identified but could lead to many product failures if manufacturing tolerances were set at the limits defined by the cluster. In order to more accurately evaluate the true robustness of a region, some idea of the value and location of the 'worst case scenario' would be useful (Parkinson *et al.* 1993). Using the concept of a 'negative' GA this system provides a novel mechanism to evaluate the robustness of regions and automatically change the definition of clusters so that they are more likely to contain good solutions.

If a GA is run inside a region defined by a cluster in a negative way (that is minimise if the primary goal is to maximise) then the value and location of the worst value in the region will be searched for and the user can evaluate the robustness of this region. If it is not good enough then the location of that minimum value (found using 'negative' search with the clustering algorithm) may indicate how limits need to be altered in order to improve the situation. Although many GA and engineering design systems allow

maximisation or minimisation of objectives, none of them appears to explicitly evaluate the quality and location of 'worst case' solutions. The negative GA will take as long as the original GA run which may be a long time (depending on the complexity of the fitness function) so it should be used sparingly. Figure 5.17 shows the result of running negative GAs inside the red (original variables) and blue (principal component) clusters. The window named Moreview5 shows the new low fitness data found inside the blue cluster, the grey-black data is that found outside the former fitness limits defined for the blue cluster. When saved to the Overview, the limits of the objective values for each cluster have been recomputed, as recommended by the warning box seen in Figure 5.17. All the data, including that found in the previous run (avoiding highlighted regions) has been saved to the Overview Window.

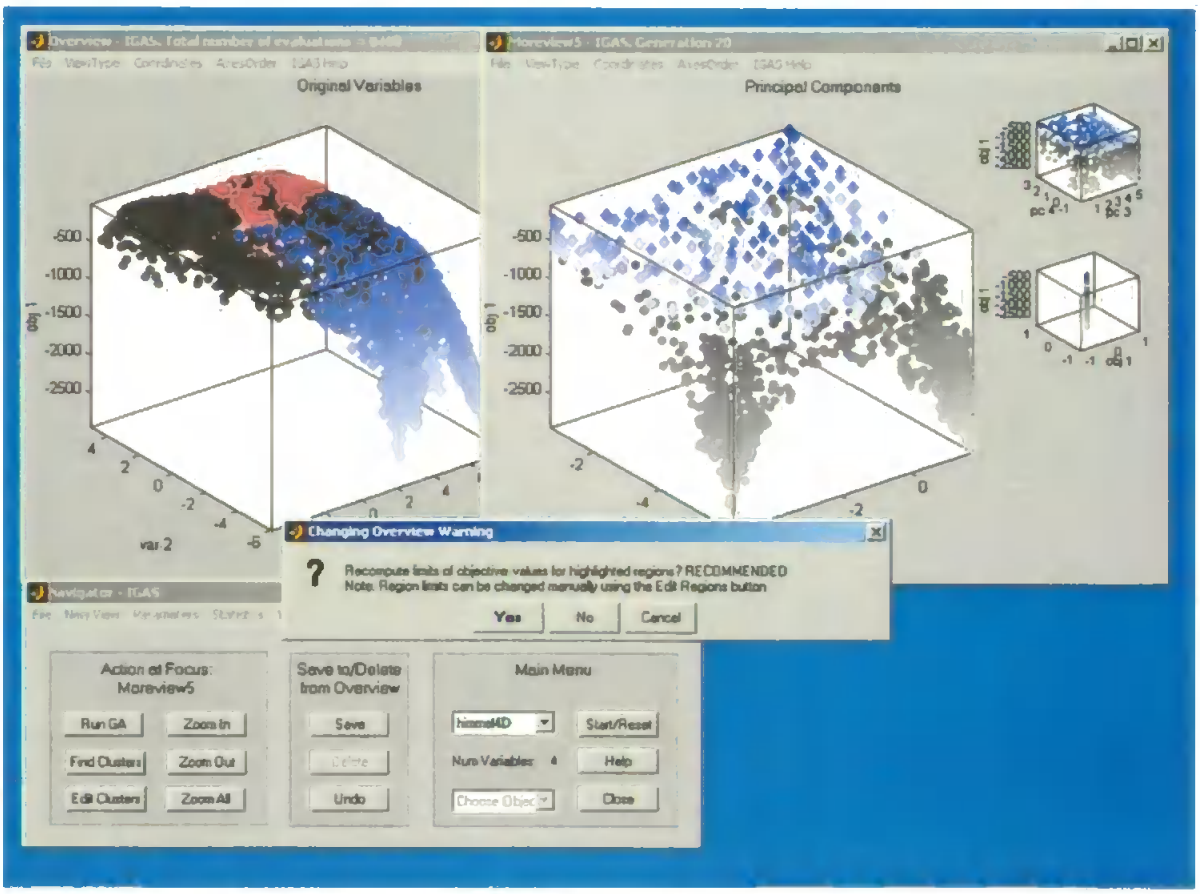


Figure 5.17: Negative GA search performed within red and blue clusters. Data can be saved to Overview Window and limits of objective values of clusters are recomputed if requested.

An important part of engineering design is the amount of tolerance that is allowed during the manufacturing process. The relative volume statistic ('Rel Vol%') can be used to compare the size of each cluster, but the difference in fitness ('Fit Ratio%') and the minimum fitness should be taken into account when evaluating the robustness of a region. In a manufacturing process there is likely to be a lower bound on the performance or objective value below which a design is not acceptable, this could be either an absolute lower bound or relative to the best solution in the cluster (local maximum). Thus it would be useful for the user to see how far away from the best solution they need to go before the fitness of the solution falls below this value. A very useful and novel part of the design system is the use of the 'Objective Filter' in the Summary of Clusters dialog box (Figure 5.18). In this example the user has set the filter at 40% of the maximum objective value found so far. When the 'Apply' button is pressed the system recomputes the limits of the highlighted regions (those with the 'Show' box checked) so that only the top 40% of the fittest solutions remain. This is reflected in the fitness ratio ('Fit Ratio%') statistic shown in the Summary of Clusters dialog; all of the values are above 60% (some more than 60% because their minimum fitness was above the global minimum before the filter was used). Further negative GAs can be performed within these new regions to see if there is further degradation of fitness, which is likely as the GA system has so far sampled a relatively small number of solutions in this continuous search space. This method is a more complete test of robustness than that of Tweedie *et al.* (1996b) (generating random solutions, then setting tolerance parameters in variable space) because bad solutions are deliberately searched for.

Clustering can also be performed negatively (see Figure 5.14a) so that the clusters containing the worst solution are chosen to show the location of the bad solutions. The user can then make a judgement on the robustness of a region by comparing the location of the fittest solution with the worst solution and adjust the limits of the region accordingly for

manufacturing purposes. Worst case analysis is a conservative way to evaluate robustness (Du & Chen 2000), so if the location or value of the worst solution compared to the best solution is such that the probability of creating the bad solution is small, then the user could choose to leave the limits as they are. An experienced engineer with extensive knowledge of the problem would make this decision, but this system allows such decisions to be made.

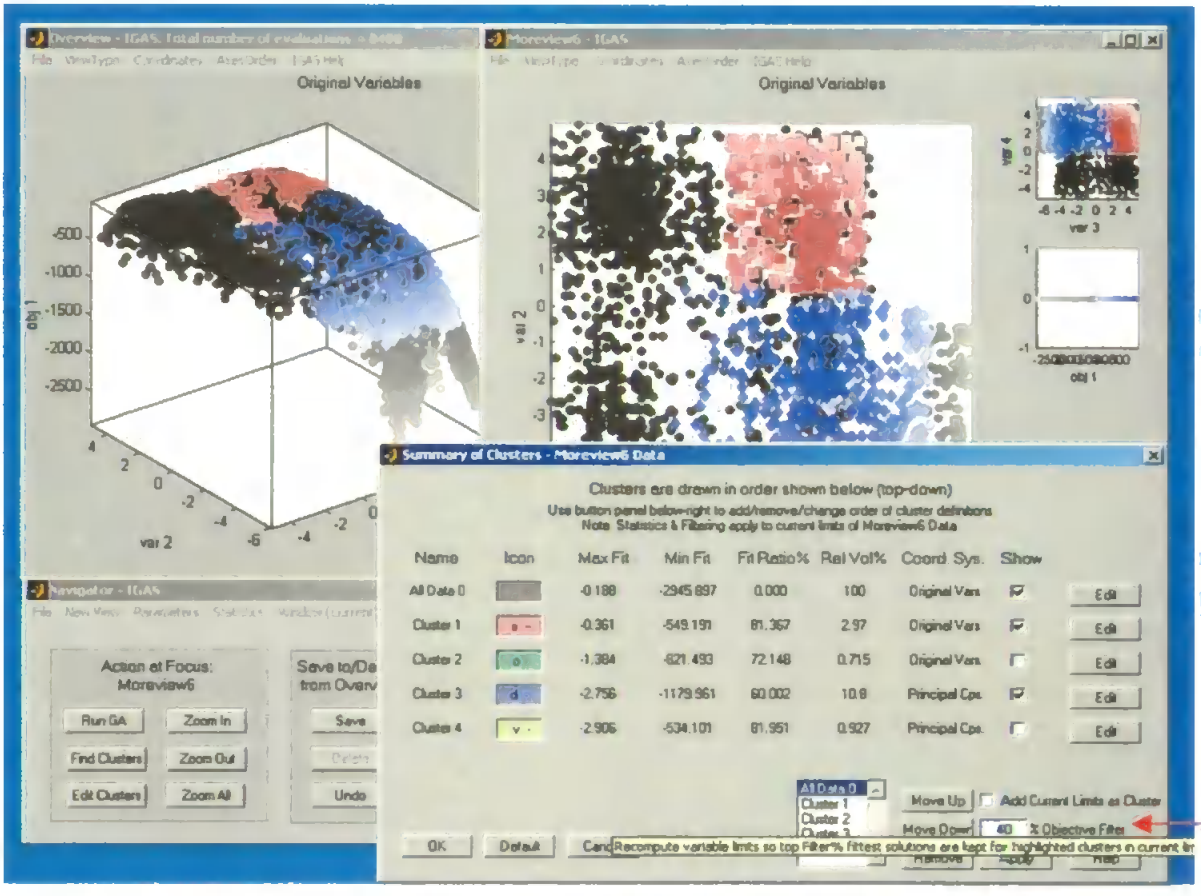


Figure 5.18: Clusters can be edited automatically using the 'Objective Filter' box (set to 40%). Only highlighted ('Shown') clusters are affected.

All the techniques described in this section will generate more data; the user can decide how much of that data shall be saved and added to the original set of solutions (any duplicated solutions will be ignored). Then recalculation of robustness can be performed to assist in further computational analysis and the user can consider the next action in the continuous design and exploration loop. The system is designed to let the user initiate

actions but to remove tedious tasks by finding obvious clusters as the search progresses. In a perfected system this would be like a cockpit design system (Spence 2003) where the system continually calculates and suggests new clusters to explore based on the user's preferences (Noy & Schroeder 2001). A novice user may not understand or want all the information produced by the computer, so this prototype system works one step at a time to avoid confusion.

The KDE-based clustering algorithm and the statistics are initially used to identify high performance regions. The user then has a choice for further exploration, either using the clustering technique to identify further clusters or running GAs to generate more data in promising regions. The negative GA helps to evaluate the noisiness of a region and locate the position of the worst solution. The objective filter lets the user define a tolerance on the allowed fitness of solutions, together with further negative runs of the GA this improves confidence in the robustness and quality of a region.

5.7 Conclusions

An easy to use and understandable interface has been implemented to help in the identification of robust regions in a mathematical or engineering design environment. The design of the interactive system follows the philosophy of Shneiderman (1998): Overview, Zoom and Details on Demand. Multivariate data can be displayed and compared using view types based on the scatterplot matrix and parallel coordinates that are easy to understand and do not distort the information. The flexible interface allows the user to manipulate the data and move around the search space, with the option to create new data and save it to the 'Overview' window if required, giving the user an overall impression of the design space. Interesting regions of the search space can either be defined using the bespoke clustering algorithm or by the user; the relevant solutions are highlighted by colour enhancing the perceived partitions in the data. The fitness (or objective value) of the

solutions is given importance by changing the perceived brightness of the data item as well as supplying a view with the fitness given in the third dimension (2D+Fitness in Z-axis). The system successfully incorporates most of the suggestions given in the literature and summarised in Chapter 3.

The automatic clustering algorithm is a novel analytical tool based on kernel density estimation (Silverman 1986) that can find the main clusters in a variety of coordinate systems. The algorithm achieves its goal of quickly supplying the main clusters in the data using very few parameters that the user does not need to know about. Clusters can be defined in the original variables or in an alternative coordinate system such as PCA and ICA that is related to the original variables. Clusters defined in alternative coordinate systems can be viewed in the original variables for comparison; again the Overview window is very handy for comparison purposes. The clustering algorithm is fairly successful on most data supplied, sometimes the results depend on the coordinate system used and eccentricities of the data (particularly small, high fitness clusters), however the user can easily change the definition of clusters if necessary. The algorithm is sufficient for the purposes described in Chapter 4, particularly for this prototype system written in MATLAB for novice users. However the technique may need enhancing to be comparable with more sophisticated clustering algorithms when knowledgeable users start to use the system.

The genetic algorithm can be used in a very flexible way, using limits defined by the user (either in original vector space or alternative coordinate systems) or the clustering algorithm to search inside and outside specified regions of the search space. The novel 'negative GA' option also lets the user check the worst case scenario inside a cluster which together with the objective filtering mechanism allows the user to set manufacturing tolerances and confirm the robustness of the region. Clusters of solutions can be compared

visually or using statistics such as the fitness ratio and relative volume enabling the user to evaluate the relative merits of regions in terms of robustness and quality. Such detailed evaluation of robustness has not been seen in before in an engineering design system based on genetic algorithms. This work extends the excellent visualisation and analysis tools of Tweedie *et al.* (1996b) by checking the worst case scenario within a region using the negative GA.

This innovative combination of genetic algorithm and clustering technique highlighted by the colouring of important clusters allows detailed exploration and exploitation of information. The use of visualisation to enhance the understanding of genetic algorithm data for engineering design is designed to allow the user to control the search process and explicitly impose their preferences on the system rather than merely waiting for the system to return 'optimal' solutions. Thus the system extends the proposals suggested by Parmee *et al.* (2000) and advocated by Mathews & Rafiq (1994); namely to support engineering design by allowing knowledge discovery through visualisation and extensive user interaction with the problem. The usefulness and success of such a system can only be confirmed with testing by human users, the following chapters describe the results of evaluation by novice users working on simulated problems and comments given by experienced engineers seeing the system at work on their own design problems.

Chapter 6: Testing Methodology: Test Functions, Evaluation of Users and Benchmark Algorithms

6.1 Introduction

This chapter describes the methodology used to examine to what extent the system helps users understand data and potentially solve engineering design problems. This preliminary investigation was designed to test the usefulness and applicability of as many features of the system as possible and compare the performance of users with GA-based 'benchmark' algorithms working on the same problems. Therefore a specific robust design task was simulated on artificial test functions allowing the system to be tested in a controlled environment; consequently the people used in the experiments were new to the problem as well as the system. In real world design the engineers may have immense knowledge of the problem and use many different ways to evaluate robustness and choose between designs (see Chapter 8). However the experimental results and critical analysis of the users working on the specific design task (Chapter 7) were instructive in assessing the usability of the system and its applicability to a general design task. This methodology provided a compromise between full psychological tests on each aspect of the system and the lengthy training of experienced engineers in using the system, both of which were outside the scope of this research.

The artificial objective functions used to simulate engineering design scenarios are described in Section 6.2. A small number of novice users tested the system's capabilities by attempting to analyse and understand those test functions using the tools on the interface. The users were asked to complete the 'engineering design task', given in Section 6.3, that involves defining robust regions of the search space according to some minimum performance level (tolerance). By comparing the hypervolume and maximum fitness of regions found they were asked to assess the relative quality of each region and provide an

order of preference. Because the relative size and shapes of peaks in the test functions were known it was possible to define the regions the user should be looking for and the relative quality of those regions according to the robustness criteria.

Section 6.4 describes the various measures that were introduced, some adapted from the evolutionary computing literature, to evaluate how well the users had understood and defined the search space. The users' quantitative assessment of regions found in each test function was obtained from a questionnaire, along with a qualitative assessment of the system and test experience. The questionnaire and further ways to assess the behaviour of the users are described in Section 6.5. The performance of the users was also evaluated by comparing their results with that achieved by three benchmark algorithms. The algorithms are traditional techniques from the multimodal evolutionary computing literature: the simple GA, GA with sharing and deterministic crowding. The relative merits and predicted performance of these algorithms are discussed in Section 6.6. The chapter concludes by highlighting the difficulty of comparing users that are attempting to complete the robust design task and algorithms that have different internal optimisation strategies.

6.2 Design of Test Functions

Test functions were designed to simulate some of the problems encountered during engineering design. The bi-objective function given in Fonseca & Fleming (1995) was converted to a multimodal function so that pertinent characteristics such as the width, height and amount of noise in each peak (or optimum) can be controlled. The characteristics of the peaks can also be defined in another coordinate system using a linear transformation matrix, this means peaks whose dimensions are correlated at an angle to the original variables can be described. Non-linear transformations are not considered here. A test function including discrete and discontinuous variables, also commonly found in engineering problems, was designed but discarded to reduce the burden on the testers; the

performance of the system working on discrete and discontinuous functions is analysed in Chapter 8.

There are two forms of the function, the first produces sparse peaks (Figure 6.1) in the search space and the second combines the definitions of peaks so that they overlap (Figure 6.2). The first function f_A is given by:

$$f_A(x_i) = \sum_p \left(\exp\left(-\sum_i \left(\frac{1}{w_{ip}} (x_i - C_{ip}) \times T_{ip} \right)^2 \right) h_p - \sum_i \left(A_{ip} - A_{ip} \cos\left(2\pi F_{ip} (x_i - C_{ip}) \right) \right) \right)$$

Equation 6.1

where: $i = 1,\ldots,n$ (number of variables)

$p = 1,\ldots,n_p$ (number of peaks)

x_i = variable values given to function

C_{ip} = centre of p^{th} peak in the i^{th} variable

h_p = height of p^{th} peak (fitness)

w_{ip} = width of p^{th} peak in i^{th} variable

T_{ip} = transformation matrix

A_{ip} = amplitude of noise

F_{ip} = frequency of noise

Domain: $0 <= x_i <= 10$

$0 <= f_A <\sim 1$

The exponential term defines the overall shape of the function whilst the second term represents noise that can be added to the function. Figure 6.1 shows the function Example_1 with the parameters given in Table 6.1.

The second function f_B is only a slight modification to the equation:

$$f_B(x_i) = \sum_p \left(\sum_i \exp\left(-\left(\frac{1}{w_{ip}} (x_i - C_{ip}) \times T_{ip} \right)^2 \right) h_p - \left(\sum A_{ip} - A_{ip} \cos\left(2\pi F_{ip} (x_i - C_{ip}) \right) \right) \right)$$

Equation 6.2

Moving the sum outside the exponential term gives very different results; the peaks are much more likely to interact with each other generating lots of data that is difficult to understand. Example_2 in Figure 6.2 has very similar parameters to Example_1, but because of the interaction, 16 peaks are generated from the two initial peaks. The exact location and width of these peaks are difficult to calculate because of the interaction (this is also true to a lesser extent for f_A), but they can be found numerically because the construction of the functions is known. The height of f_B is sometimes bigger than one (the maximum height of an individual peak) because of the interaction between peaks.

Test functions based on these constructions were given to the users for evaluation; Table 6.1 shows the parameters used (see Appendix B for transformation matrices used in Test_2). Figures 6.3 - 6.5 show the test functions with each peak coloured down to approximately 50% of the local optimum. The test functions vary in complexity and difficulty of locating peaks.

Function	No. Vars.	Func. Type	No. Peaks	Centre (per var.)	Height	Width (per var.)	Noise Amp./Freq.	Alt. Coord.
Ex_1 (Fig 6.1)	4	f_A	2	[3 3 3 3] [7 7 7 7]	1 .5	[.5 .5 .5 .5] [1 1 1 1]	[5]/[1] [10]/[1]	No
Ex_2 (Fig 6.2)	4	f_B	2(16)	[3 3 3 3] [7 7 7 7]	1 .5	[.5 .5 .5 .5] [1 1 1 1]	No	No
Test_1 (Fig 6.3)	5	f_A	4	[3 3 3 7 7] [7 7 3 3 3] [2 5 2 5 2] [8 5 8 5 8]	1 .75 .5 .25	[1.5 1.5 ...] [3 3 3 3 3] [1.5 1.5 ...] [3 3 3 3 3]	[5]/[1] [5]/[1] [5]/[1] [5]/[1]	No
Test_2 (Fig 6.4)	4	f_A	3	[3 3 3 3] [7 7 7 7] [2 8 2 8]	0.5 1 .3333	[3 1 3 1] [3 1 3 1] [3 1 3 1]	No	No 45deg 30deg?
Test_3 (Fig 6.5)	4	f_B	2(16)	[3,3,3,3] [7,7,7,7]	0.25 1	[3 3 3 3] [1.5 1.5 ...]	No	No

Table 6.1: Parameters for examples and test functions (see also Appendix B).

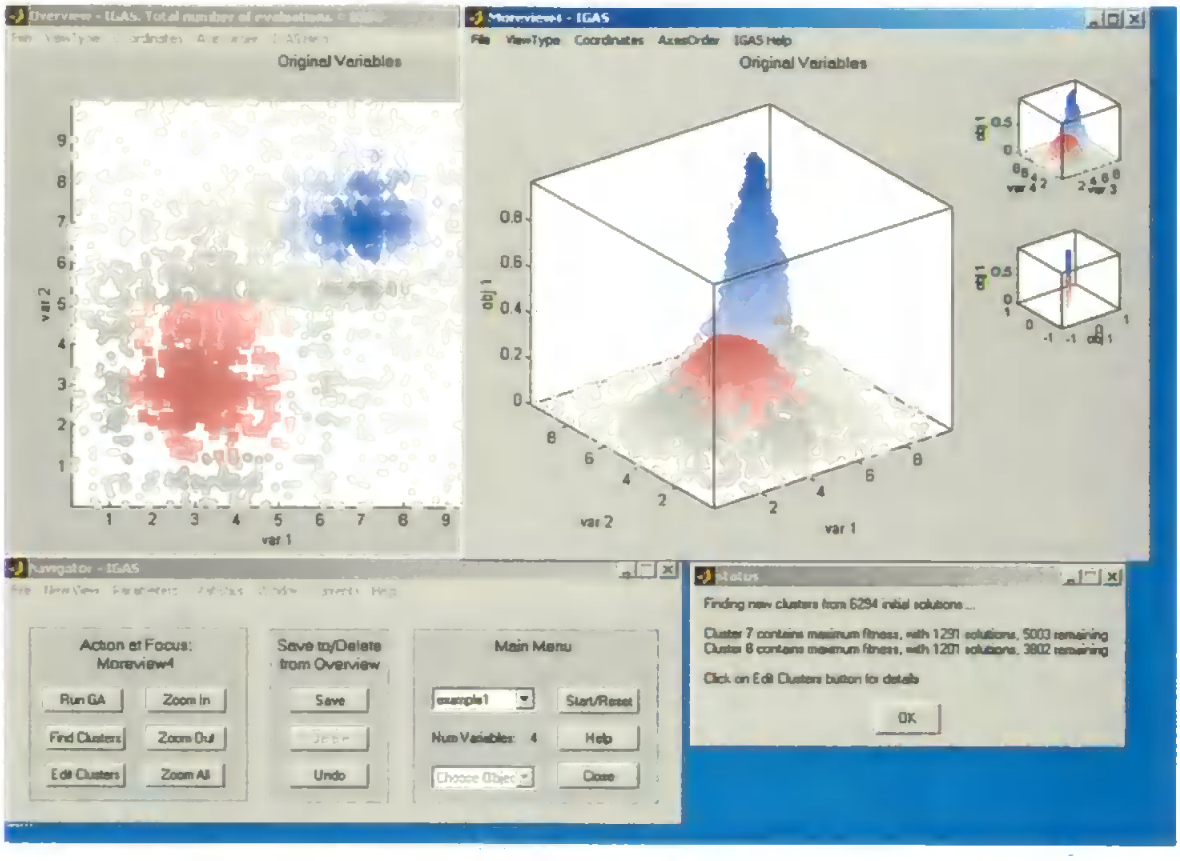


Figure 6.1: Example_1 (function type f_A): peaks with different widths/height/noise.

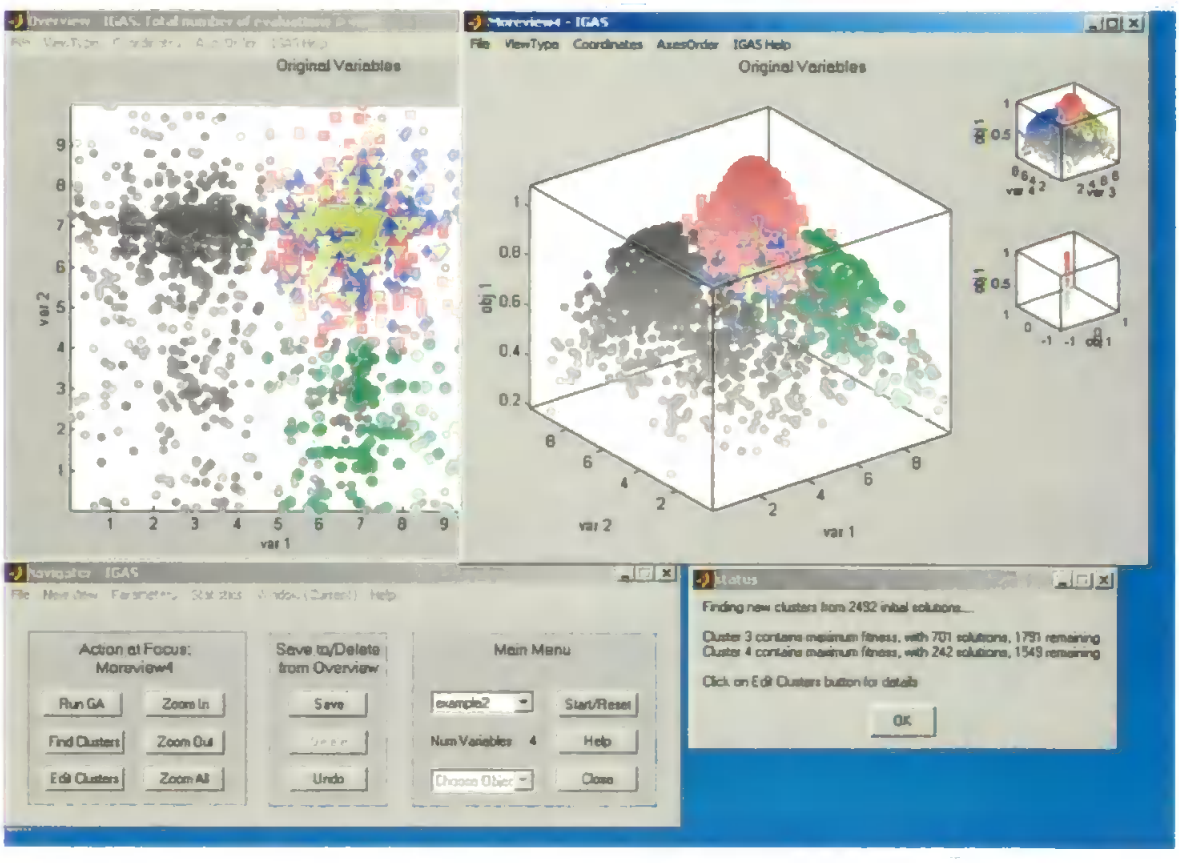


Figure 6.2: Example_2 (function type f_B): many possible peaks, difficult to understand.

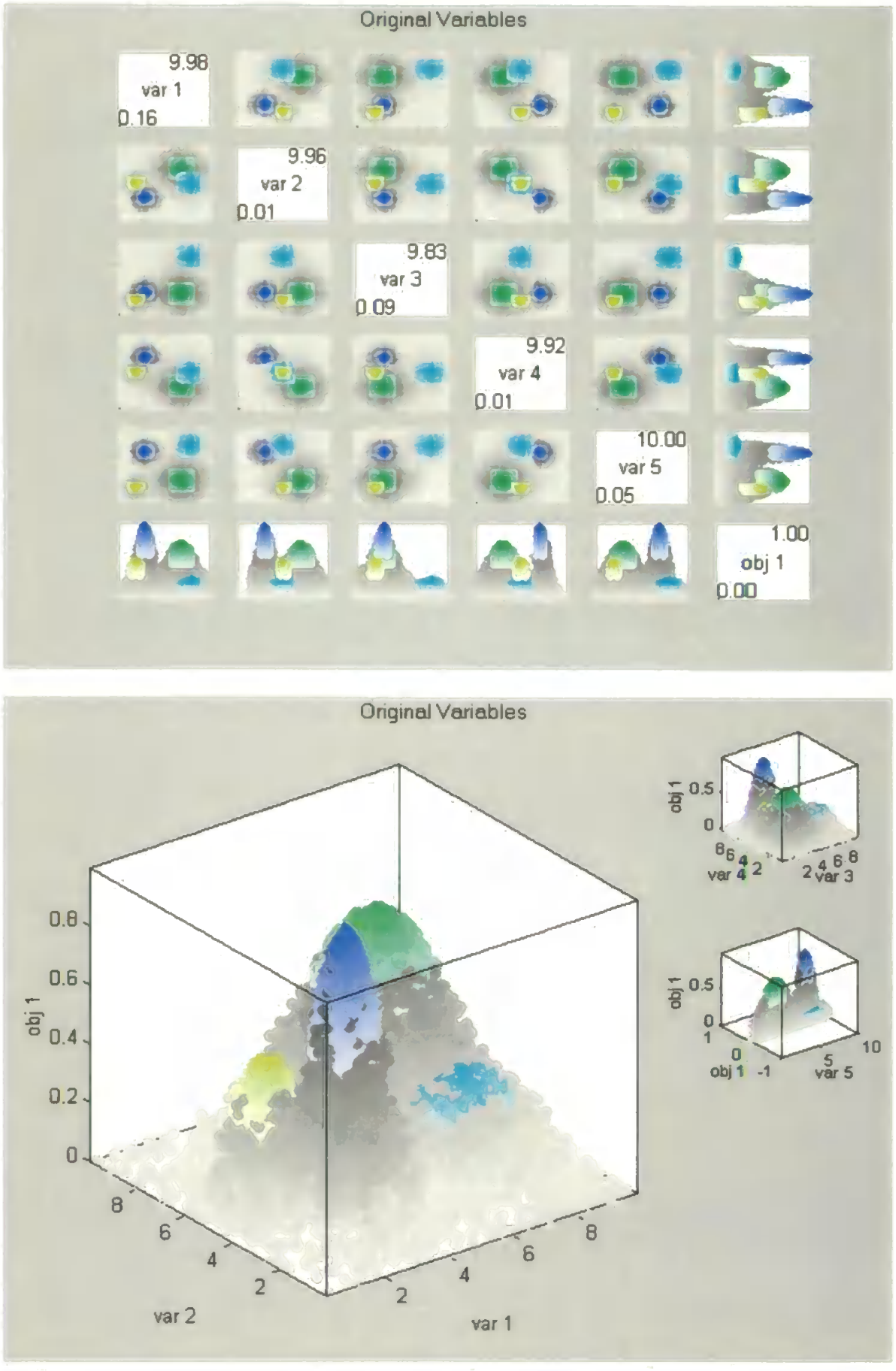


Figure 6.3: Test_1, peaks defined in original coordinate system with some noise, two of the peaks small and very hard to find. The scatterplot matrix (above) and 2D with fitness in the z-axis (below) views are shown.

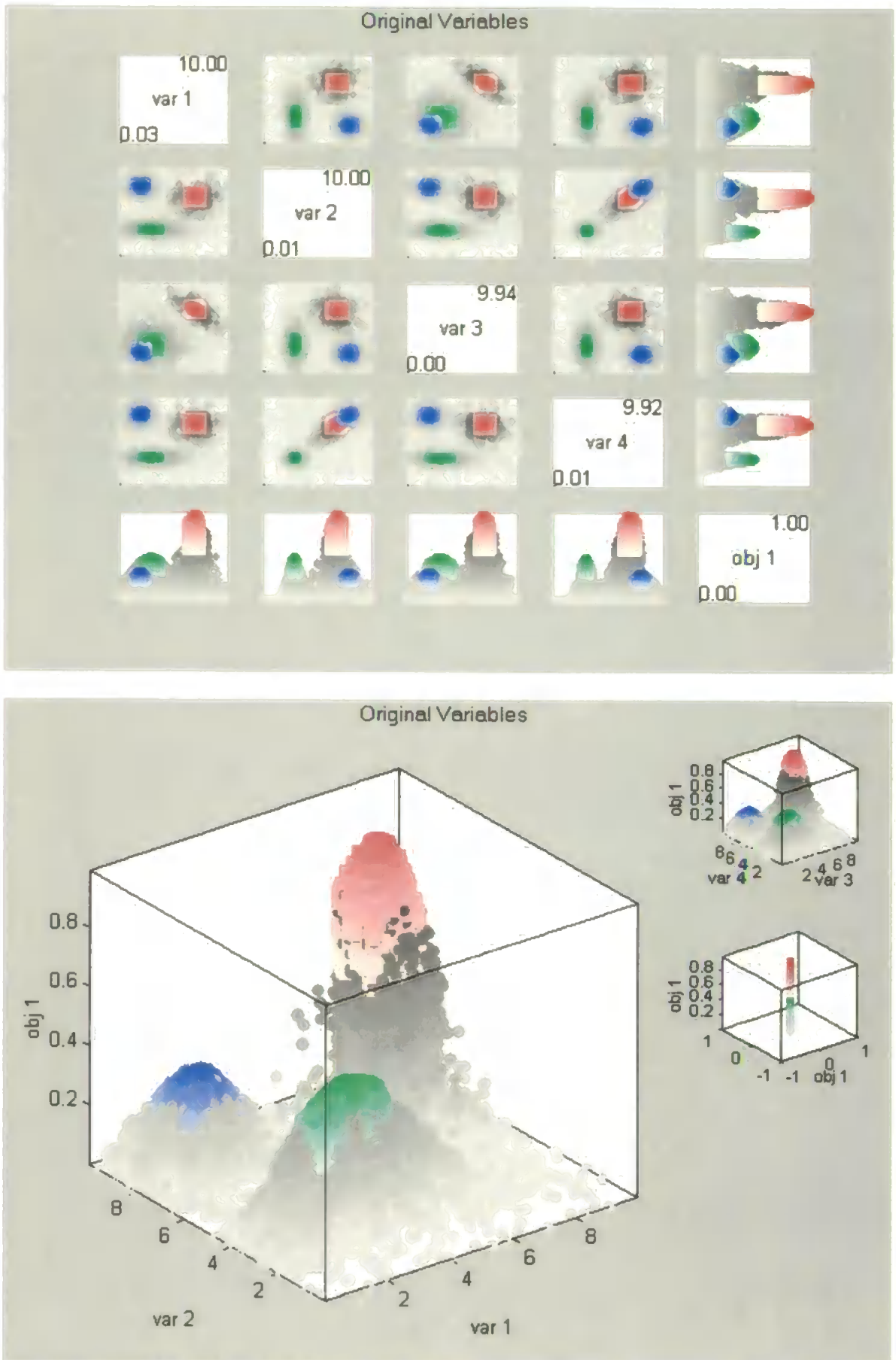
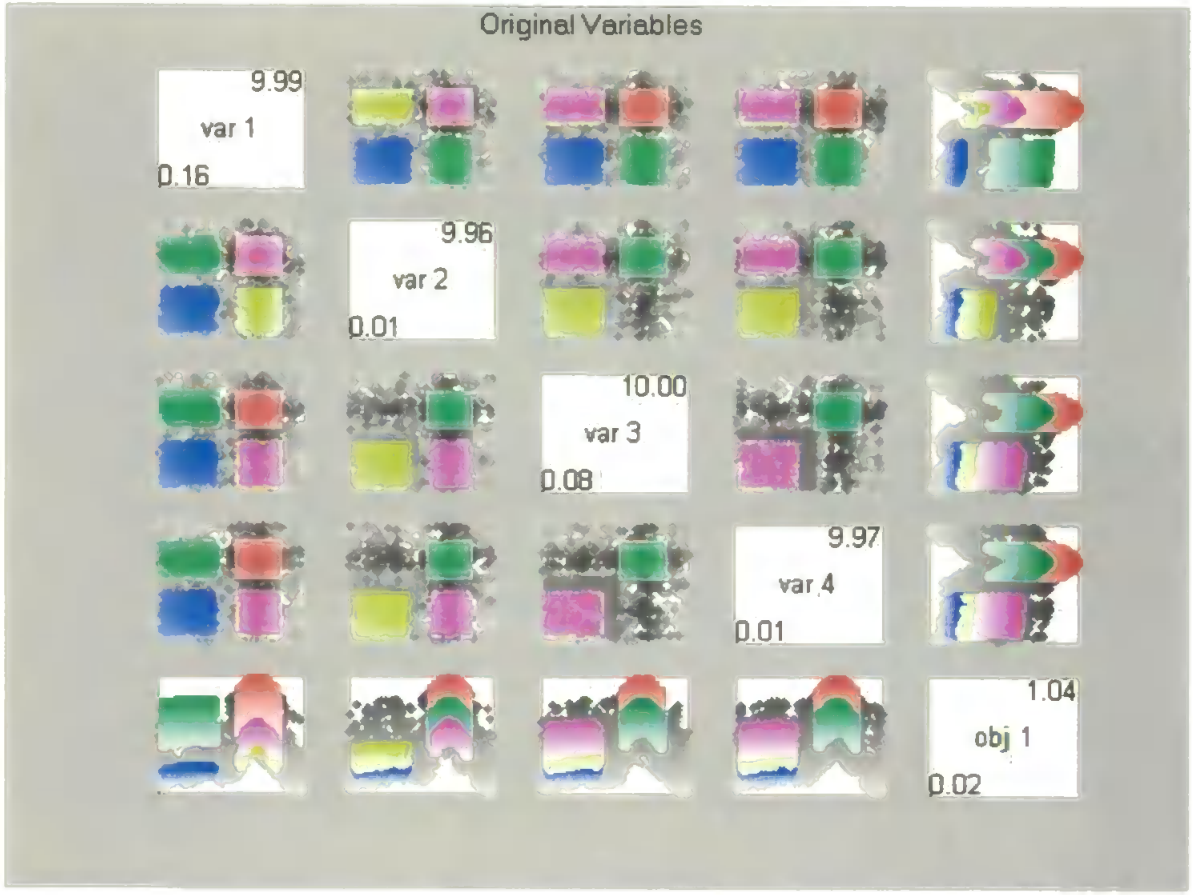


Figure 6.4: Test_2, two of the peaks defined in alternative coordinate systems, easier to find but difficult to define boundaries.



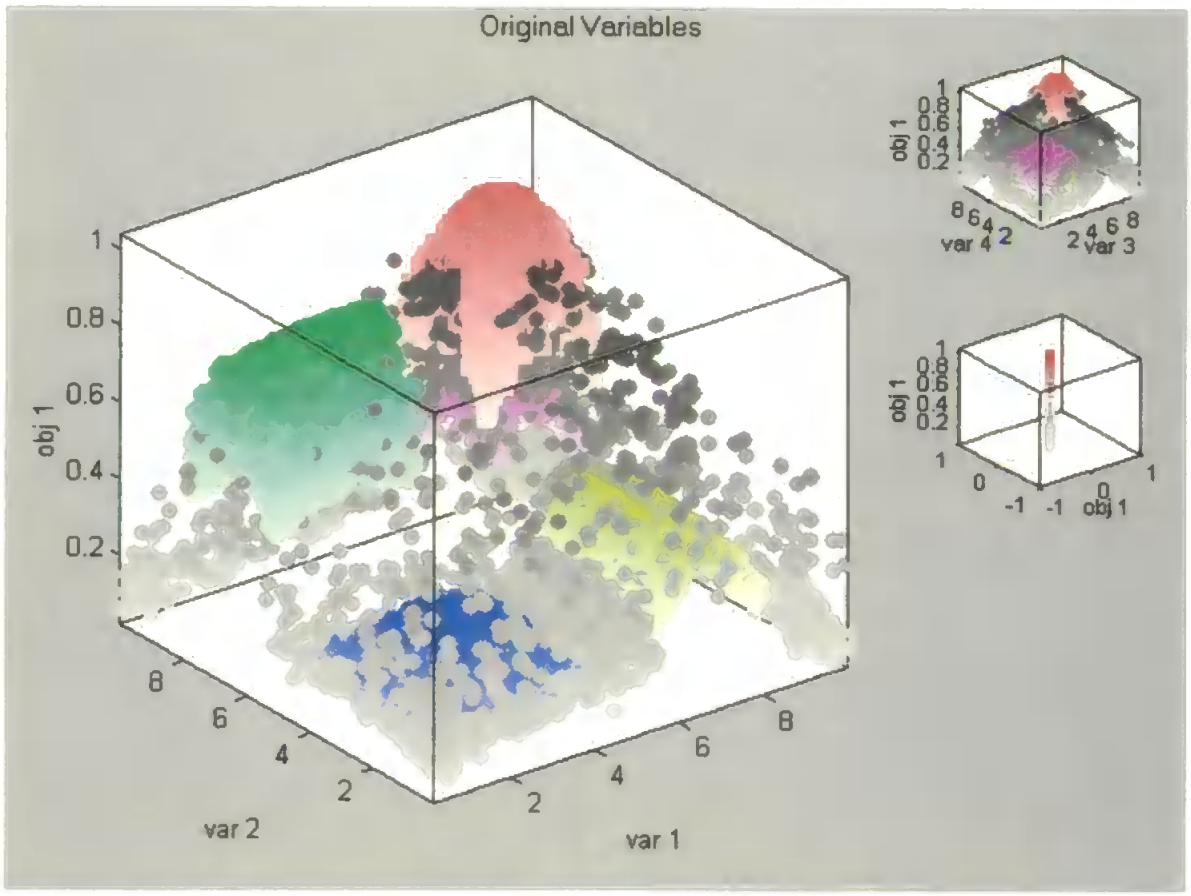


Figure 6.5: Test_3, defined using f_B in the original coordinate system. Five of the sixteen peaks highlighted in colour.

6.3 Robustness and Quality of Regions Found

To simulate an engineering design scenario using the system a definition of high performance regions and the quality of those regions was needed. Such a definition would enable the user to evaluate the relative quality of regions and thus clarify their goals during the engineering design task, it would also enable the construction of 'ideal' regions to evaluate user performance and validate their conclusions.

For most engineering design problems the quality is individually defined, so the exact definition of robustness and quality cannot be defined across problems. However the definition of quality is usually related to some performance value (such as minimising cost or attaining a target value) whilst ensuring that the design meets some manufacturing specifications (Taguchi 1986, pp. 13-21). The definition of the problem will determine whether these manufacturing specifications are given in the form of a tolerance on the performance (objective), parameters (variables) values or both. Either set of tolerances could be given in the form of a hard constraint (for example 'definitely not more than') or soft constraint ('as close as possible to'). Tweedie *et al.* (1996b) define the tolerances in variable space and iteratively change them to agree with objective criteria. Phadke (1989) gives examples of trying to meet different types of manufacturing tolerances set in objective space (*ibid.* pp. 15-16) and changing parameters simultaneously to achieve tolerances set in both variable and objective spaces (*ibid.* pp. 28-29). Either approach will achieve the aim of defining robust or non-sensitive regions that are as wide as possible in variable space but cause minimal changes in objective space, although they may return different answers.

In these user evaluation experiments the robust design task was set up to use as many of the features currently available on the system as possible. Therefore the tolerance was set in objective space and the users were encouraged to find regions in variable space

that agree with the tolerance. The regions can then be evaluated and compared in terms of maximum objective value and size of the acceptable region in variable space, encouraging decision making that is a major part of engineering design.

For an objective function $f(x)$, the quality of an optimum is given by:

$Q = M * V$ where Q = Quality of region,
M = Relative value of optimum,
V = Relative hypervolume of shape at tolerance level T.

For simplicity V is defined thus:

$V = \prod_{i=1:n} t_i$ where n = Number of variables.
If m_i = Position of optimum in i^{th} variable
and T = Tolerance level, find largest $2*n$ sided shape containing m such that $f(x) > T$,
then t_i = Length of the i^{th} side of the shape.

In the following example of **two** variables the **tolerance** required T is 50% of the optimum. The line defining the tolerated region is a circle, so the **largest** 4-sided **shape** that fits inside the circle is a square. (M' and V' are actual values of optimum and hypervolume; normalise to find M and V relative to overall limits).

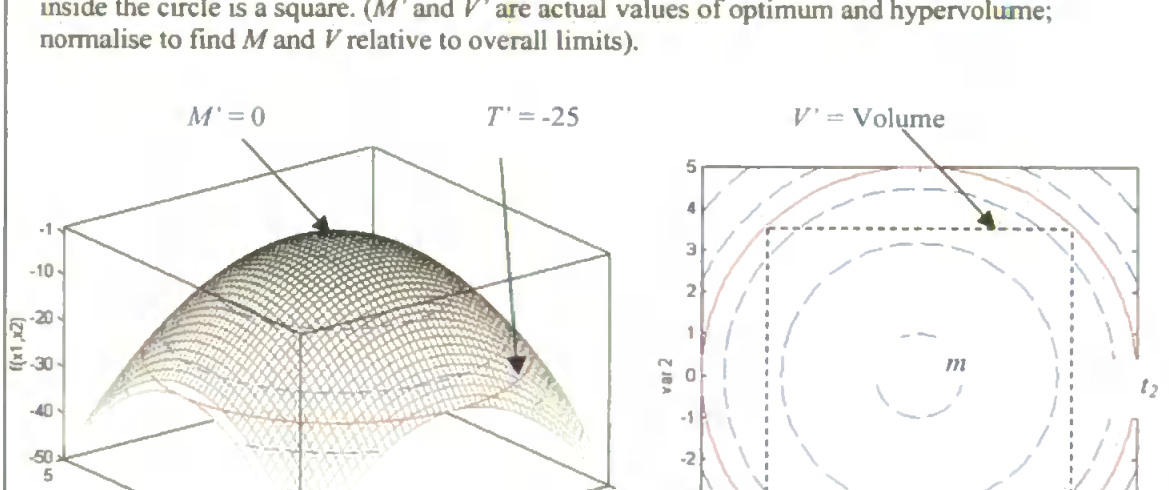


Figure 6.6: Definition of Quality given in engineering design task.

The definition of quality including robustness based on tolerance in objective space is illustrated in Figure 6.6. For any given optimum found in the search space, the aim is to find a region such that all solutions inside the region have a value within a certain tolerance level of the *local* optimum. For simplicity the region of acceptability (or robust region) is

defined as the largest 2*n sided shape that fits inside the tolerated region. The quality is then found by multiplying the fitness value of the local optimum by the hypervolume of that region. A tolerance level of 50% from the local optimum was set in objective space as acceptable; here the word tolerance was used to define the amount of change in objective space, that in turn defines the tolerated (feasible) region in variable space. The coloured regions shown in Figures 6.3-6.5 are the nearly ideal regions that obey this definition. The fitness and volume of the regions found by the users were compared against these ideal regions.

Table 6.2 shows the quality measure for each ideal region derived from the actual fitness and hypervolume values and their corresponding normalised values, Appendix B gives details of how these regions were numerically defined taking the interaction of peaks into account. Quality values for the third test function are particularly intriguing; the height and hypervolumes for the peaks are inversely proportional, but this quality value gives more importance to the volume. This is possibly the opposite to how the function is perceived by an observer – the difference in height is more apparent than the difference in width. Test_2 also implies that the hypervolume of peak 3 is the highest, which does not appear so when viewing Figure 6.4, however these regions are defined in alternative coordinate systems, so their true volume is different to that perceived in original coordinates. The normalised quality value is used to give an approximate rank for each peak related to the other peaks in that function, where 1 is low quality and 5 is high quality (last column in Table 6.2).

Func.	Peak No. (colour)	Height (actual)	Hypervol (actual)	Quality (actual)	Height (norm)	Hypervol (norm)	Quality (norm)	Rank (1-5)
Test_1 (Fig 6.3)	1 (blue)	1.001	.000012	.000012	.398	.011	.022	2
	2 (green)	.750	.000538	.000404	.298	.510	.740	5
	3 (yellow)	.516	.000011	.000006	.205	.010	.010	1
	4 (cyan)	.251	.000495	.000124	.099	.469	.228	3
Test_2 (Fig 6.4)	1 (green)	.500	.000432	.000216	.273	.278	.246	1
	2 (red)	1.000	.000432	.000432	.545	.278	.492	5
	3 (blue)	.333	.000693	.000231	.182	.444	.262	2
Test_3 (Fig 6.5)	1 (blue)	.251	.0151	.00362	.0242	.1123	.0482	1
	2 (yellow)	.449	.0109	.00471	.0434	.0816	.0627	4
	3-5	"	"	"	"	"	"	
	6 (magenta)	.647	.0080	.00494	.0625	.0594	.0658	5
	7-11	"	"	"	"	"	"	
	12 (green)	.845	.0058	.00471	.0816	.0432	.0626	3
	13-15	"	"	"	"	"	"	
	16 (red)	1.043	.0042	.00422	.1008	.0315	.0563	2

Table 6.2: Actual Fitness, hypervolume and quality measures for each test function, with normalised values. The colour of the peak is that shown in the relevant figure number. The rank for each peak is related to the normalised quality value; *1 is low* and *5 is high*. In Test_3 there are 16 peaks of five different types, Figure 6.5 highlights an example of one at each level. See also Tables B.1 and B.2 in Appendix B.

The definition given in Figure 6.6 provides a quality measure for every peak in the test functions. However when a user is using the system some data may look more robust than it actually is because not all the search space has been sampled, therefore there is an issue with apparent robustness of a region due to the shape of a peak and its true robustness found by sampling the whole region and analysing noise inside the peak. Although this statistic gives an absolute value, the quality of a peak from a user or engineer's point of view depends on the relative importance given to the fitness or hypervolume of the region. For an engineering problem the person who designed the function can assess this, but in the testing process it was left to the perception of the user.

The full engineering design task given to the user before the experiments is supplied in Appendix C.2. The users were asked to achieve a tolerance level of 50% of fitness of a local optimum, however to fully complete the engineering design task the users

needed to compare regions by defining their width in all variables as wide as possible without including infeasible individuals (with fitness less than 50% of the local optimum). So the implied engineering design task is to meet a hard constraint on the edge of that constraint which is a very difficult problem. To evaluate the extent users have understood the search space and engineering design task they are asked to give a rank for the relative quality of regions they have found. It is likely the users will not find all the peaks, so cannot be expected to duplicate the exact rank, but their results should reflect the order of the ranks given here.

6.4 Measuring Criteria

6.4.1 Methodology

McGraw (1992) suggests the testing of an interface should investigate the features the system was designed for, so the aim of the experiments was to assess the effectiveness of the system in helping users perform the engineering design task. This involves combining optimisation and clustering techniques, so the effectiveness of those techniques was assessed as well.

The aim of the experiments was to give an indication of how the system aids decision making, as well as the exploration attributes of the joint GA and clustering techniques. Given the definitions of height and hypervolume of each peak, can the users find the correct regions and use the system to make a choice about the comparative quality of each region? As the quality of a region is given as a trade-off between the height and width, they can only give subjective answers. Measures are needed to check if the correct regions have been found and defined, has the user searched for good fitness solutions and checked the robustness? Do they check that a region is robust before going onto the next? Have they spent too much time searching in bad areas of the search space?

The results need to be compared against the benchmark techniques from the evolutionary computation literature that are used to solve the problem of multimodal fitness functions. Advantages and disadvantages of the benchmark algorithms are well known (see Sections 2.3.3 and 6.6) so are ideal for comparing with a new system and drawing conclusions. However most GA-based methods are designed to just find local and global optima within a function, whereas the engineering design task requires users to assess the size and robustness of the region around each local optimum. GAs will find local optima but not necessarily define the region without outside help. To solve the problem of comparing very different types of algorithm working on the same problem a compromise between the engineering design and evolutionary computing approaches was needed.

In the same way that the definition of quality and robustness will be different for most engineering problems, there are also no definitive guidelines on how to evaluate a system for engineering design because they are mostly designed to solve individual problems. The evaluation of how well a decision maker has performed is also undertaken on a case by case basis and no generic testing procedure has been produced. In the evolutionary literature most researchers suggest a new system and show the results on low dimensional test functions then give some new results to a complicated engineering design problem, without understanding the problem there is no way of knowing if better or more robust solutions exist. No obvious testing procedure has been put forward for more complicated artificial test functions.

Therefore metrics from the evolutionary literature are adapted to assess how closely users and algorithms define the desired regions of the search space given in Section 6.3 (Table 6.2). Two sets of quite similar measures are described in the next sub-section: the first assess how close the definition supplied by the user is to the true definition; the second set of measures compares the actual data produced by users and algorithms with the

correctly defined regions. The second set of measures allows comparison between the users and algorithms and indicates occasions when users have found desired regions without realising it.

6.4.2 Analysis of User-Defined Regions

Three metrics have been implemented to compare the *regions* defined by the user to the ideal regions. They are independent and in the range 0-1, so the product of the three gives an overall quality measure for each experiment.

Metric 1: Correct Ratio C^R

This measure assesses the number of peaks found by the user and correctly identified by the user. The most common statistic in multimodal optimisation experiments is the success rate of optima found in a function (Beasley *et al.* 1993), which can be expressed as a ratio of the number of correctly identified peaks to the number of actual peaks. But when users are defining regions themselves they may identify two clusters on a single peak; this behaviour is penalised by including the ratio of number of correctly identified peaks to the total number of regions identified by the user. So the Correct Ratio C^R is given by:

$$C^R = \frac{(N_c)^2}{N_a . N_d} \qquad \textit{Equation 6.3}$$

where N_a = number of actual peaks in the search space

N_c = number of correctly identified peaks

N_d = total number of regions identified by the user as peaks

Metric 2: Fitness Accuracy F^A

The second metric is also commonly used in multimodal optimisation experiments; the accuracy of the best fitness found by the user to the nearest true local optimum. This is given as a ratio of the best fitness to the actual fitness:

$$F^{A} = \frac{1}{N_d} \sum_{r} \frac{B_r}{F_l} \qquad \textit{Equation 6.4}$$

where $r = 1,\ldots,N_d$ (number of regions identified by user)

l = a member of $1,\ldots,N_a$ (number of actual peaks in the search space)

B_r = best fitness in each region r identified

F_l = fitness of the closest true local optimum l to region r

Metric 3: **Hypervolume Accuracy H^A**

The third metric is similar to the second but measures the accuracy of the volume of the region defined by the user. The hypervolume defined by the user for each identified region can be compared with the hypervolume of the nearest robust region (the ideal region that satisfies the tolerance requirement given for each peak in a function):

$$H^{A} = \frac{1}{N_d} \sum_{r} \begin{cases} \frac{U_r}{R_l} \; if \; U_r \le R_l \\ \frac{R_l}{U_r} \; if \; U_r > R_l \end{cases} \qquad \textit{Equation 6.5}$$

where $r = 1,\ldots,N_d$ (number of regions identified by user)

l = a member of $1,\ldots,N_a$ (number of actual peaks in the search space)

U_r = hypervolume of each identified region:

$$U_r = \prod_{v} \left[\max(x_v^r) - \min(x_v^r) \right] \quad (x^r \text{ is the space defined by } r \text{ in each variable } v)$$

and R_l = hypervolume of the ideal region l closest to each r:

$$R_l = \prod_{v} \left[\max(x_v^l) - \min(x_v^l) \right] \quad (x^l \text{ is the space defined by } l \text{ in each variable } v)$$

When the actual peaks are defined in an alternative coordinate system the hypervolume will be defined in this system. For fair comparison the data identified by the user (in any coordinate system) is transformed to the actual peak coordinates and the resulting hypervolume compared to the volume of the nearest robust region.

The hypervolume accuracy metric compares the actual size of the regions defined to the size of the nearest local optimum, the location of the regions is not defined. It is possible that the user may define a region that does not correspond to the true local optimum and yet happens to have the same hypervolume as the required region. In most cases the fitness accuracy statistic will be worse in this case, so in combination the fitness and robustness accuracy statistics gives a good indication of how well the user has defined the region.

Combined User Metric:

The overall metric for user definitions of regions is a product of the three metrics defined here:

$$O^U = C^R.F^A.H^A \qquad \textit{Equation 6.6}$$

The product of these terms means that all three metrics need to be high (where 1 is the maximum) to give a good overall value of O^U.

6.4.3 Analysis of All Data Found During Session

In this section metrics to evaluate the *data* generated by the user or an algorithm are described. The definition of regions set by the user are ignored, instead the sampling of ideal regions defined by the true peaks are assessed. These metrics allow any algorithm or user/algorithm combination to be compared. The values are again given in the range 0-1 and can be combined in an overall metric for each problem.

Metric 4: **Maximum Peak Ratio M^R**

The success and accuracy of peaks found by the algorithm are defined in one metric called "maximum peak ratio" (Miller & Shaw 1996). For each peak the best (fittest) solution that falls inside an actual peak region (that is defined at 50% of each local maximum) is

compared against the local maximum. The maximum peak ratio for all peaks is the average of each local peak ratio:

$$M^R = \frac{1}{N_a}\sum_i \frac{B_i}{F_i} \qquad \textit{Equation 6.7}$$

where $i = 1,\ldots,N_a$ (number of actual peaks in the search space)

B_i = best fitness found by user in each region i (if none found $B_i = 0$)

F_i = actual optimal value of each peak i

If no solution is found within the peak region, that peak will have a 'zero peak ratio, conversely if maximum peak ratio $M^R = 1$, all peaks and their exact local optima have been found.

Metric 5: **Inner Hypervolume Accuracy I^A**

Metric 5 is the hypervolume of all solutions that have been found inside each ideal region compared to the actual hypervolume. For each peak this value will always be less than or equal to 1. The average of the hypervolume ratio for all peaks identified gives the inner hypervolume accuracy:

$$I^A = \frac{1}{N_c}\sum_j \frac{U_j}{R_l} \qquad \textit{Equation 6.8}$$

where $j = 1,\ldots,N_c$ (number of correctly identified peaks)

l = a member of $1,\ldots,N_a$ (number of actual peaks in the search space)

U_j = hypervolume of data inside each correctly identified region j:

$U_j = \prod_v \left[\max(x_v^j) - \min(x_v^j)\right]$ (x^j is the space defined by region j in each variable v)

and R_l = hypervolume of the local region l closest to each region j:

$R_l = \prod_v \left[\max(x_v^l) - \min(x_v^l)\right]$ (x^l is the space defined by region l in each variable v)

Again for actual peaks defined in alternative coordinate systems all the data will be transformed to that definition and the subset of data that falls inside the peak definition will be compared with the true definition.

Metric 6: **Productivity Ratio P^R**

This metric calculates how much search has been concentrated in the desired regions. Any solutions that fall outside those regions are deemed as redundant. However solutions at the bottom of a peak may have been used to find the good solutions, therefore it is not expected that this metric will achieve a value of 1, but will tend towards it. The amount of productive search is defined as the ratio of solutions found inside actual regions to the total number of all solutions found by user or algorithm:

$$P^R = \frac{\sum_i n_i}{n^T} \qquad \textit{Equation 6.9}$$

where $i = 1,\ldots,N_a$ (number of actual peaks in the search space)

n_i = number of solutions found inside each actual region i

n^T = total number of solutions found during an experiment

Combined Data Metric:

The overall metric for the quality of the data produced by an algorithm or system is again the product of the three metrics defined above:

$$O^D = M^R . I^A . P^R \qquad \textit{Equation 6.10}$$

Metrics 5 and 6 together give an indication of how well the true regions have been sampled. For each region if the inner hypervolume accuracy is large and the number of solutions is large it is assumed that the sampling across the region is good. If the inner hypervolume metric is large but the number of solutions is small, the sampling is not good, but this will be reflected in the productivity metric. If the size of the inner hypervolume

metric is small this indicates that only a small part of the region has been sampled anyway. These metrics do not indicate the efficiency of the sampling, for example if nearly the whole region has been identified but most of the solutions are concentrated in one part of the region, the metrics will be unchanged. Additionally measuring the sampling efficiency is not fair on multimodal algorithms that are not designed to search around local optima, they just try to find individual peaks. For a restricted set of experiments these metrics are sufficient as a compromise that measures the ability to find and define regions but no more.

6.4.4 General Measures and Discussion

Other measures to enable comparison between runs are the number of evaluations and time spent on each problem.

Metric 7: **Number of Evaluations**

For all experiments a maximum number of evaluations of 30 000 was set. This is not a very high number, although could take a significant amount of time if the evaluation function was very complex. For the benchmark algorithms this number of evaluations is always used. But the users were told to use as many of these evaluations as they wanted and often they were happy enough with their work to finish before using them all, or became tired and disinterested so finished early. Comparisons can still be made despite this anomaly.

Metric 8: **Time**

The amount of time spent by the users and for different algorithms on each problem is given. The time includes CPU time by the genetic algorithm and clustering technique as well as thinking time by the user. The thinking and interaction time (including redrawing of figures) spent by the humans overshadows the CPU time of the algorithms, but is instructive nevertheless.

One measure often used during multimodal experiments that has not been mentioned is the chi-squared measure used by Deb & Goldberg (1989) to test the sharing algorithm. They asserted that the number of solutions generated on each peak should be proportional to the fitness of the peak and used a chi-squared like measure to ensure the distribution is correct. In contrast to traditional functions, the functions used in these experiments have additional characteristics such as varied hypervolume of peaks and amount of noise, so an alternative proposal could be used such as generating the number of solutions proportional to the hypervolume or even the quality measure defined in Section 6.3. Such a proposal could be seen as matching the 'relative effort' of the user (number of solutions generated) to the size or quality of the peak. However this was not an explicit aim of the engineering design task set for the users, nor is it the aim of all multimodal algorithms, so it is unfair to set such a target.

These measures show how many peaks have been found and to what extent they have been explored without going into too much detail. The combined effect of the overall measures allows this evaluation without biasing the results towards engineering specific tasks or multimodal optimisation algorithms. The number of peaks found and accuracy of fitness and robustness is expected to be higher for the user driven experiments. Another subjective element of the testing is to ask whether the user has learnt more about the search space whilst interacting with the data. This can be partially answered by statistically comparing their evaluation of the search space with the theoretically ideal definition of quality, the users also need to be asked if they understood the engineering design task and found the system useful in attempting to solve it.

6.5 User Evaluation and Questionnaire

A small group of four people volunteered to undertake this preliminary evaluation of the system without recompense, they were peers of the author with varying amounts of knowledge of engineering design and evolutionary algorithms, none of them had used the system before but all were given a demonstration of the system during a seminar and a further demonstration at the start of the test. The users were novices with the system and the problem with limited time available so the author was on hand to answer any technical questions and to provide assistance for the more complicated operations. Users were encouraged to discuss the system and explain how they were solving the problems verbally, the author recorded their comments and their actions were recorded to a data file by the system. Participants were given the following materials that are shown in the Appendix C:

1. Introduction to the System and Tests (C.1)
2. The Engineering Design Task (C.2)
3. Picture of the two-dimensional Himmelblau Function (C.3)
4. Questionnaire for each Test Function (C.4)

The questionnaire was designed for participants to fill in at the end of each session (Appendix C.4). They were encouraged to identify the regions of the search space they considered important and label them as coloured clusters using the features of the interface. In Question 1 they are asked to rank the regions by giving a quality or preference measure, this allows a comparison between the actual quality of regions found and the perceived quality by the user, it was not expected that they should give the exact rank but give an order that reflects the true quality given in Table 6.2. They were then asked to rate their confidence in the quality measures given to get some indication of how happy they are with the answers they have given and work done. Question 3 asks how certain they were

that all peaks were found and in Question 4 the users were asked to rate the usefulness of the system. Questions 1-4 were answered with a number between 1 (low) and 5 (high). Finally Question 5 was an open question to encourage comments and constructive criticism about the system.

As well as the formal feedback from the questionnaire it was possible to extract information about how users tackled the task and what features they used to complete the tasks through their comments and the recorded actions. Of particular interest in this study were the following algorithms and features:

- Preferred view type (e.g. 2D Scatter / Parallel Coordinates...)
- Use of alternative coordinate systems (PCA/ICA...)
- Zoom and multiple views used and understood?
- Order of axes changed (AxesOrder dialog) to compare different variables?
- GA parameters changed or diversity algorithms used?
- Use of 'negative' GA
- Clustering algorithm parameters changed?
- Clustering algorithm used or clusters defined manually?
- Use of 'negative' clustering search
- Objective Filter used or another procedure followed to redefine regions?
- Use of colours to identify regions of the search space
- Exploratory behaviour – how did users balance the trade-off between concentrating on particular regions and searching for new regions
- Save to Overview used?
- Summary of clusters used to differentiate between regions?

This preliminary investigation was used to evaluate the system as a whole and get general feedback on the usefulness of the features in helping to solve the engineering design problem. A lot of information was gained from the users in the discussions during the tests, however it is accepted that more formal methods are needed to evaluate particular attributes of the system. These experiments were designed to allow the users to make decisions with fewer restrictions by simulating an engineering design environment. The definition of quality and resulting 'ideal' regions of the search space also allows an evaluation of the influence of the user on the genetic algorithm search.

6.6 Comparison of User Performance with Evolutionary Algorithms

Having identified the peaks in each objective function and determined the characteristics of each associated high performance region, it is possible to evaluate the performance of traditional evolutionary algorithms and diversity techniques and compare with user performance. These algorithms do not 'label' regions so Metrics 1-3 of Section 6.4.2 cannot be used, but the other metrics allow comparison of the number of peaks found and how much they have been investigated. The algorithms chosen were well known benchmark algorithms and tested diversity techniques:

1. The simple GA
2. GA with Sharing (Deb & Goldberg 1989)
3. Deterministic Crowding (Mahfoud 1992)

Parameters for these algorithms are shown in Table 6.3. The simple GA uses crossover and mutation to both explore the search space and exploit any good solutions that are found. However the simple GA will soon converge on a good solution and the power of selection will encourage the population to be almost identical, the only way to a new peak is by random mutation to a very good individual, which is unlikely. The default

system parameters used by the participants in the experiment are the simple GA running for 20 generations with the mutation scheme added to ensure no duplicate solutions (Section 4.3.1 and Figure 5.12b); the mutation scheme slows down the algorithm (especially as number of generations increases) but ensures some local diversity.

The sharing algorithm encourages diversity by reducing the fitness of solutions that are close to other solutions according to some pre-defined notion of 'closeness' (Section 2.3.3). Sharing will have an advantage over the users because the number of peaks will be given to it, but the peaks are not evenly spaced in Tests 1 and 2 so this advantage may be cancelled out. Another disadvantage of the sharing algorithm is the time complexity incurred during the distance comparison between individuals.

Algorithm	Parameter Name	Parameter Value / Type
Simple GA	Number of Generations	300
	Populations Size	100
	Number of Bits per Variable	16
	Elitism	None (100% replacement)
	Selection Type	Stochastic Universal Sampling
	Crossover Type	Double Point
	Crossover Rate	.7
	Mutation Rate	.7 (/nbits/nvar)**
Sharing* (reduce fitness)	Sharing Parameter σ_{share}	Depends on number of peaks, and size of search space
	Alpha	1
Deterministic Crowding* (DC)	Selection Type	Replacement Selection (used in DC algorithm)
Default System in User Experiments*	Number of Generations	20
	Duplicate Mutation Rate	5 (/nbits/nvar)**

*As simple GA parameters unless otherwise specified
**nvar=number of variables, nbits = number of bits per variable

Table 6.3: Benchmark algorithm parameters. See Section 2.3.3 for further details of the Sharing and DC algorithms.

Mahfoud (1992) developed deterministic crowding (DC) that does not use any parameters (beyond those used by the simple GA) and has no extra computational cost (Section 2.3.3). Diversity is maintained by only keeping offspring if they are similar to their parents, so DC tends to maintain peaks at a rate proportional to their width (or hypervolume) although dominated peaks will be lost (Mahfoud 1994). Mahfoud (1995) devised an important set of experiments that compared the sequential niche and sharing techniques to parallel diversity algorithms such as DC. He showed that on most problems the crowding technique was more likely to be successful because multiple optima can be maintained naturally in the population.

It is acknowledged that other multimodal techniques such as CHC (Eshelman 1991), the ECO GA (Davidor 1991), the sequential niche technique (Beasley 1993), dynamic sharing (Miller & Shaw 1996) and multi-population GAs such as the forking GA (Tsutsui & Fujimoto 1993) are more successful on certain functions than the traditional diversity techniques described here. All these algorithms achieve differing results on different functions because of the characteristics of the algorithm and an in-depth empirical study could be followed to show the best set of techniques on these particular functions. However this thesis is an investigation of the benefit of human-computer interaction in evolutionary computation and the visualisation and analysis of the output. So the basic algorithms are used for comparison here knowing that more informative information could be generated using algorithms that are better designed.

To allow comparison with the user evaluations, each stand alone benchmark algorithm was allowed 30 000 function evaluations per experiment, the average of 10 experiments are reported. There are four participants in the user experiments who can only perform each test once to avoid domain knowledge improving results. The results of the user experiments will naturally have a much higher variance than the algorithms; they may

also choose not to use all the evaluations available to them. Experiments in the literature usually present the results of the final generation to see if the algorithm has maintained all peaks, even though many of the peaks can be seen in some functions by just viewing a random selection of data points in the search space. By default the system keeps all the data found during a GA run in the user experiments, so to be fair all the data was kept and analysed for the comparison algorithms also, meaning that peaks lost by the algorithm are kept in the final analysis.

6.7 Conclusions

The experiments described in this chapter differ from traditional multimodal comparisons in a number of ways. Most researchers use test functions that are either simple, low dimensional problems (Deb & Goldberg 1989) or multidimensional problems with regularly spaced peaks such as the Griewangk and Rastrigin functions (Gordon & Whitley 1993). Instead special artificial functions have been created to simulate data found in engineering design scenarios; multidimensional and difficult to understand, the peaks are not evenly spaced and have different widths and heights with noise that makes the robustness of regions even more unclear. Many engineering design problems contain discrete variables or discontinuities in the fitness function; such characteristics were not included in these theoretical experiments to reduce the complexity for the users, but they are considered in the real world case studies of Chapter 8.

To evaluate the performance of users and algorithms the number, height and width of the peaks in each test function was found. The robustness of a peak was defined as the hypervolume at 50% fitness from the local optima. Combining the hypervolume and fitness statistic gives a good trade-off value for the quality of a region. Measuring criteria were easily formed based on the hypervolume and fitness to evaluate the number of real peaks found and how well a user defines the high performing regions using the interface

provided. The performance of multimodal benchmark algorithms can also be compared with user results by analysing the data produced using similar statistics as well as assessing the productivity of the user or algorithm (the number of solutions found in good regions).

The system evaluators were also asked to fill in a feedback questionnaire to discover their perception of the search space. They were asked to give a rank for the quality or relative importance of regions found, this information can be compared with the theoretical ranking based on the quality statistic. Such a comparison between the users' perception and reality will clarify how much the system helped the participants understand the problem. The subjects were also asked an open question to return their criticisms and suggestions about the system. The author recorded verbal comments and actions made during the tests were recorded by the system providing additional feedback. Users' behaviour such as interface features used and ways of tackling the problem were of interest, although the complexity of the task and system meant that not all features would be used.

These experiments are concerned with evaluating the quality of each peak and to encourage the user to explore as much as possible and compare with the behaviour of benchmark algorithms. For multimodal algorithms the most important criterion is a 100% success rate, that is find all the peaks in the search space, but the algorithms try to solve the problem using different characteristics of the data; sharing will distribute solutions in relation to the fitness of peaks found whilst DC uses the proportionate width of the base of peaks. However the goal of the engineering design task set for these experiment is to allow the user to evaluate both the fitness and width of peaks, so it is hypothesised that the users will achieve the most consistent results in all metrics. Chapter 7 documents the results of the user evaluation and benchmark comparison experiments and analyses the behaviour and comments returned by the users.

The system was designed for novice users with knowledge of the problem who will become expert users as their understanding of the system improves (Section 3.2.3), whilst expanding their knowledge of the problem. The quantitative evaluation of such a system working on a general engineering design problem is difficult because of the individual requirements of the problem and the many different ways robustness can be defined (some of these difficulties are revealed in the initial case studies of Chapter 8). As a compromise these experiments provide some insight into the success of combining the genetic algorithm with novice users on a specific task they are not familiar with, a critical analysis of the experiments will reveal the potential of combining the human and evolutionary computing to general design situations. The analysis will also indicate the clarity and usefulness of the engineering design task that will provide further understanding of how to define robustness.

Chapter 7: Results of Evaluation of System by Users and Algorithms

7.1 Introduction

This chapter presents results of the preliminary experiments performed on the artificial test functions presented in Chapter 6. The chapter is organised as follows: Section 7.2 reports on the performance of users in defining the correct regions defined in Chapter 6 and the results from the actual data returned compared to the true regions. The diversity of educational background and experience of the participants is highlighted in Appendix D (Table D.1). With such a small number of testers it is impossible to make definite conclusions from their performance, the complexity of the system and task to be completed also makes a purely statistical analysis even more difficult. Therefore in Section 7.3 an empirical analysis of questionnaire results about the tests is followed by a descriptive analysis and discussion of individual comments made by the users. The participants' results are compared with the benchmark algorithms from the evolutionary literature in Section 7.4. Again statistical analysis of this comparison was very difficult due to the small number of users and complexity of the problem. This preliminary study was not designed to provide statistical proof of the relative performance of the system, but the results are used to make comments on how characteristics of the algorithms affect the search compared to user behaviour. A qualitative assessment of user actions and feedback provided more illuminative and constructive conclusions. A critical analysis of the quantitative and qualitative results, the engineering design task, the experimental set up and the system itself is given in Section 7.5. Lessons learned from the analysis suggested some automatic procedures and alternative ways to define robustness that are described in Section 7.6. Conclusions are drawn in Section 7.7.

7.2 Summary of Online User Results

7.2.1 Introduction

Two sets of results are presented for the user evaluation, the first set evaluates the actual regions defined by the users, the second set evaluates the data generated by the user compared to the target regions as described in Chapter 6 (Tables 6.1 and 6.2). The author (known as User_0) having expertise of the system and domain knowledge is included as a reference to compare with the performance of the novice participants (Users 1 to 4). The statistics derived for User_0 are taken from the data and regions displayed in Figures 6.3 to 6.5 (note that five regions were identified in Test_3 for clarity). Some visual examples of the results returned by the participants are shown in Figures 7.1 to 7.4 at the end of Section 7.2, these are referred to later in the text.

7.2.2 User Metric Results

Table 7.1 shows the analysis of the user-defined regions compared with the true regions. It is immediately obvious from the "correct ratio" statistic that the number of peaks identified varies a lot from function to function, indicating the diverse nature of user behaviour and background. However once a peak has been identified, the best fitness is usually found fairly close to the true best ("fitness accuracy"). Some users spent more time and energy trying to find the correct width (reflected in the "hypervolume accuracy" measure) than others who may have misunderstood the problem or did not find the width to be an important part of the decision making process. User_2 correctly identified all three peaks for Test_2, but did not identify the size of them very well, conversely User_3 was the worst at finding peaks in each test function, but was generally better at defining the size of the regions. This is reflected in the "combined user metric" for Test_1; because the product is used, a much higher hypervolume accuracy for User_3 results in a higher combined metric. The fact that other users found more regions should perhaps be given more weight in the combined metric.

Function (Num. Orig. Peaks)	User Name	No. Peaks Identified | Correct	Correct Ratio	Fitness Accuracy	Hypervol. Accuracy	Combined User Metric
Test_1 (4)	**User_1**	**3 | 2**	**.333**	**.938**	**.078**	**.024**
	User_2	**2 | 2**	**.5**	**.987**	**.109**	**.054**
	User_3	**1 | 1**	**.25**	**.985**	**.533**	**.131**
	User_4	**3 | 2**	**.333**	**.988**	**.108**	**.036**
	User_0	4 | 4	1.000	.996	.912	.909
Test_2 (3)	**User_1**	**5 | 2**	**.267**	**.723**	**.323**	**.062**
	User_2	**3 | 3**	**1.000**	**.997**	**.053**	**.053**
	User_3	**3 | 1**	**.111**	**.948**	**.239**	**.025**
	User_4	**2 | 2**	**.667**	**.987**	**.266**	**.175**
	User_0	3 | 3	1.000	.997	.380	.379
Test_3 (16)	**User_1**	**6 | 5**	**.260**	**.966**	**.421**	**.106**
	User_2	**4 | 2**	**.063**	**.924**	**.677**	**.039**
	User_4	**4 | 4**	**.25**	**.999**	**.187**	**.047**
	User_0	5 | 5	.313	.999	.967	.302

Table 7.1: Analysis of regions defined by Users 1 to 4. For each user the number of Identified Peaks and of those the number of Correct Peaks is given. The resulting Correct Ratio, Fitness and Hypervolume Accuracy, and the Combined Metric are given as defined in Section 6.4.2. 'User_0' is the author using domain knowledge.

Generally the combined user metric indicates the difficulty of the test functions. Test function two is the easiest to solve, especially in identifying the right number of peaks, because there are only three peaks and each one is easily seen in the search space (in fact the bottoms of all peaks are available after the first run of the GA, but there is no evidence that any of the users spotted this). Their width is defined in alternative coordinate systems and cannot be correctly specified in the original coordinate system, as evidenced by the low hypervolume accuracy relative to fitness accuracy for Test_2. Tests 1 and 3 appear to be equally difficult, although for different reasons. Test_1 has few peaks but two of them are particularly difficult to find because they are so small or 'hidden' by other peaks (the location of the peaks are similar in some parameters). Test_3 has many peaks that are easy to find and users do define the ones they find very well, but they do not bother to identify more than about five, the other peaks are not considered important.

7.2.3 Data Metric Results

Table 7.2 shows the analysis of the actual data produced by the user during each experiment revealing some different results to the analysis of the user-defined regions. The metrics also differ from those shown in Table 7.1. The "maximum peak ratio" takes into account both the number of peaks and accuracy of the maximum fitness of the peaks. "Inner hypervolume accuracy" is measured on any data found inside the true regions as defined in Table 6.2. "Productivity Ratio" is the number of solutions found inside the true regions compared to the total number of solutions generated; the users were not explicitly asked to improve this characteristic, but generally get a good value as the nature of the task required the users to generate solutions in good regions. User_4 in particular spent more time producing solutions within the true regions, improving the productivity ratio and thus the "combined data metric" compared to the explicit analysis of Table 7.1.

In many cases users actually found peaks without identifying them during the session (compare number of peaks found in Table 7.2 with number of peaks correctly identified in Table 7.1). This reveals that a lot of the important information is found by the system, but there is a failure by the user to realise the importance of that information and failure by the system to report it. Test_3 appears to have the worst results in this regard; on average less than 4 peaks were correctly identified out of almost all 16 peaks available. However many of the peak-defined regions contained a small number of solutions that would be difficult to spot due to the complex and difficult to understand nature of the search space. Most of the users identified the fittest peak in Test_3 and up to four of the next fittest peaks that are of equal fitness, then considered all the rest of the data as noise. This is a perfectly reasonable assumption to make if the user has an absolute level of fitness in mind below which solutions should not be considered and for all functions the users appeared to use this assumption, despite the design task asking them to consider local fitness and hypervolume to define quality.

Function (Num. Orig. Peaks)	User Name	No. Peaks Found	Maximum Peak Ratio	Inner Hypervol. Accuracy	Product. Ratio	Combined Data Metric
Test_1 (4)	**User_1**	**2**	**.474**	**.455**	**.112**	**.024**
	User_2	**4**	**.936**	**.556**	**.099**	**.052**
	User_3	**2**	**.478**	**.741**	**.158**	**.060**
	User_4	**2**	**.495**	**.735**	**.589**	**.215**
	User_0	4	.996	.876	.214	.186
Test_2 (3)	**User_1**	**3**	**.814**	**.262**	**.363**	**.077**
	User_2	**3**	**.998**	**.972**	**.289**	**.280**
	User_3	**3**	**.869**	**.338**	**.287**	**.084**
	User_4	**3**	**.929**	**.500**	**.298**	**.139**
	User_0	3	.997	.967	.254	.245
Test_3 (16)	**User_1**	**15**	**.898**	**.336**	**.555**	**.167**
	User_2	**16**	**.987**	**.800**	**.587**	**.463**
	User_4	**16**	**.971**	**.698**	**.758**	**.514**
	User_0	16	.964	.498	.463	.222

Table 7.2: Analysis of the data produced by the users. The number of actual peaks found is given along with the Maximum Peak Ratio (combining number of peaks with maximum fitness of peaks), Inner Hypervolume Accuracy and Productivity Ratio. The Combined Data Metric defined in Section 6.4.3 is the product of the latter three. 'User_0' is the author using domain knowledge.

Function (Num. Orig. Peaks)	User Name	No. Evaluations	Real Time (s)
Test_1 (4)	**User_1**	**31500**	**3353**
	User_2	**31500**	**4693**
	User_3	**25200**	**2026**
	User_4	**18900**	**3766**
	User_0	33600	1516
Test_2 (3)	**User_1**	**31500**	**3013**
	User_2	**31500**	**2870**
	User_3	**14700**	**2000**
	User_4	**16800**	**1851**
	User_0	25200	5735
Test_3 (16)	**User_1**	**31500**	**2837**
	User_2	**21000**	**1647**
	User_4	**23100**	**2250**
	User_0	35700	6100

Table 7.3: Number of evaluations used and time in seconds (s) spent by each user on the test functions. 'User_0' is the author using domain knowledge

All users found all peaks in Test_2, but obviously produced a better inner hypervolume accuracy if they knew they had found them. However User_2, who identified all the peaks, had a low productivity ratio. This is because a lot of the solutions were only defined correctly in two of the variables indicating that this user concentrated on just two dimensions and did not take into account the other two dimensions. These factors mean that many solutions were generated outside the true regions, although the true regions were identified well. Solutions outside the true regions help to define the peaks, so it is desirable that this happens (a productivity ratio of 1 would indicate less exploration).

User_4 produced very good overall metrics for all test functions and seemed to learn how to solve the problem the best (defining regions and generating solutions inside them to check) despite knowing little about engineering design. User_1 had the worst overall performance which was surprising for the author as this participant was known to understand the engineering design problem very well and spent a long time on each function (on average around 52 minutes). This participant often identified two or more regions on the same true peak and spent a long time analysing those regions and generating low fitness solutions to ensure the definitions were correct. There was noise present within the true peaks, so it could be argued that a number of regions exist on each peak; this behaviour was maybe closer to that of true engineers, although worse results were obtained because the ideal regions were wider than those identified.

7.2.4 General Metrics

These results indicate varying success by the users in understanding the problem and using the system to solve the problem. A large amount of variation is due to the complexity of the problem and different ways the engineering design task and quality measure was interpreted. The time taken to complete the tasks and number of evaluations used also have a large variance (Table 7.3), although the participants completed the tasks faster as they

learned the system. Not all users used all the 30 000 evaluations available to them, they were either satisfied with the results achieved or did not want to continue out of tiredness or boredom. If anything there is an inverse correlation between the amount of time or evaluations used and the results obtained, indicating that more time spent does not necessarily mean finding new regions or obtaining better statistics. Indeed the author (User_0) spent the most amount of time trying to define the exact regions in Test_3, obtaining worse results than most of the other users. The author's results indicate the difficulty of the task even with domain knowledge.

Due to the small number of participants available to undertake these experiments, no conclusions can be made from a statistical analysis of the results. However a qualitative or reflective analysis of the users' actions and comments made during the tests is much more useful (Bucciarelli 1984, Schön 1984). The following sections include such analysis.

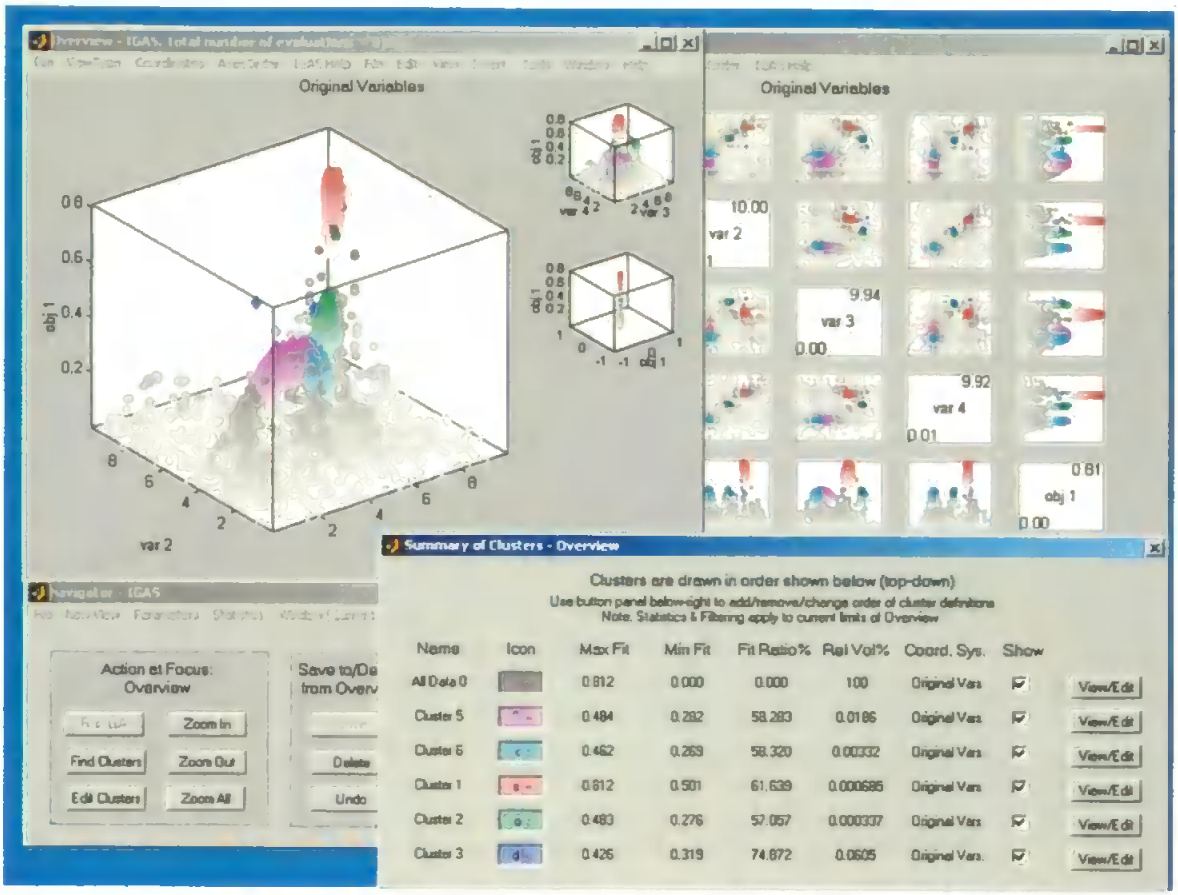


Figure 7.1: Example of User_1 final result (Test_2).

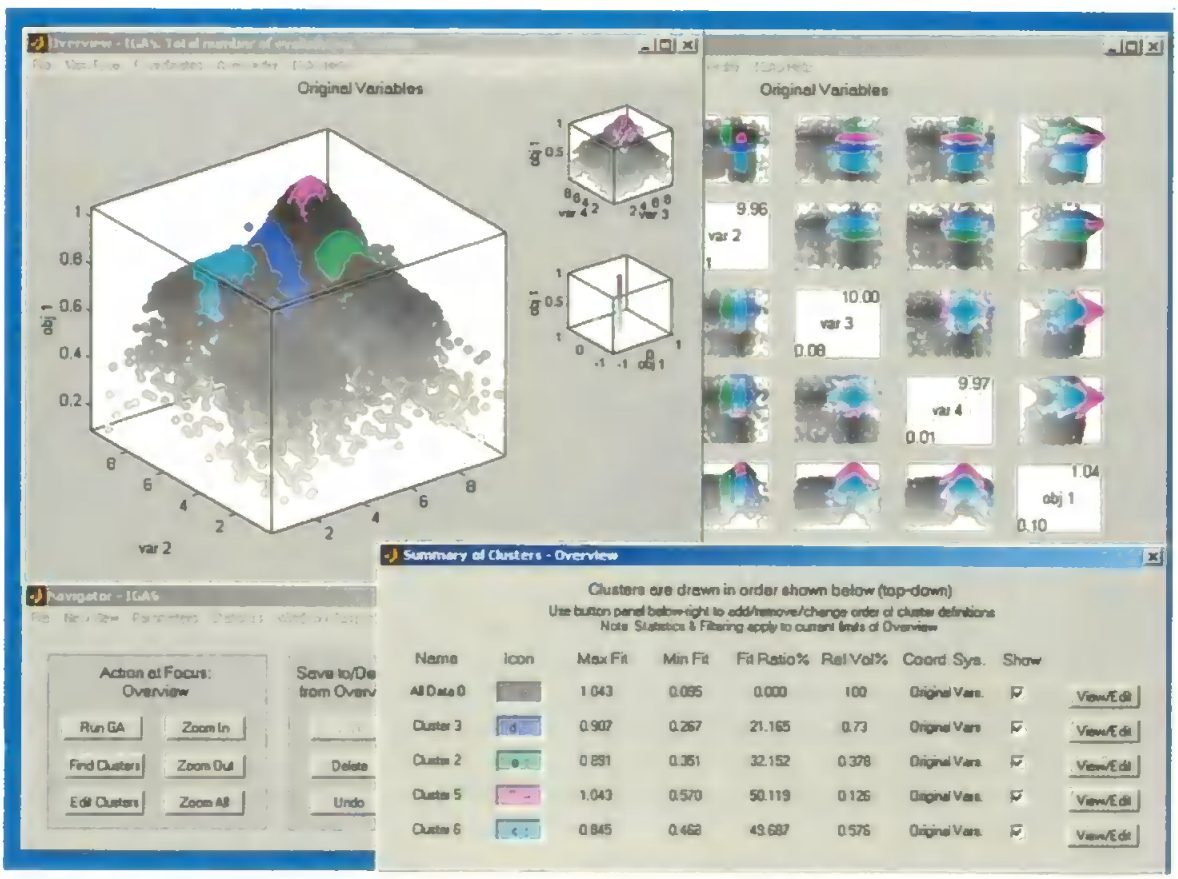


Figure 7.2: Example of User_2 final result (Test_3).

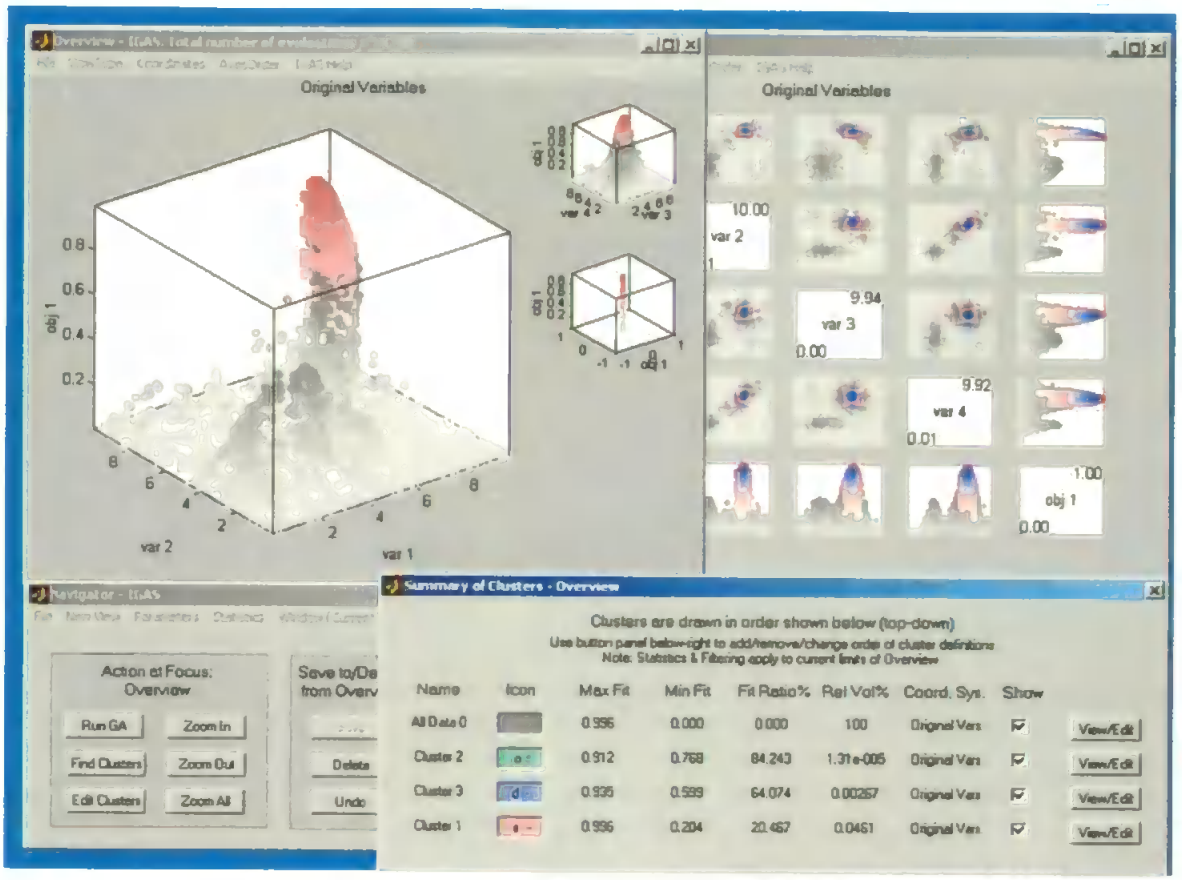


Figure 7.3: Example of User_3 final result (Test_2).

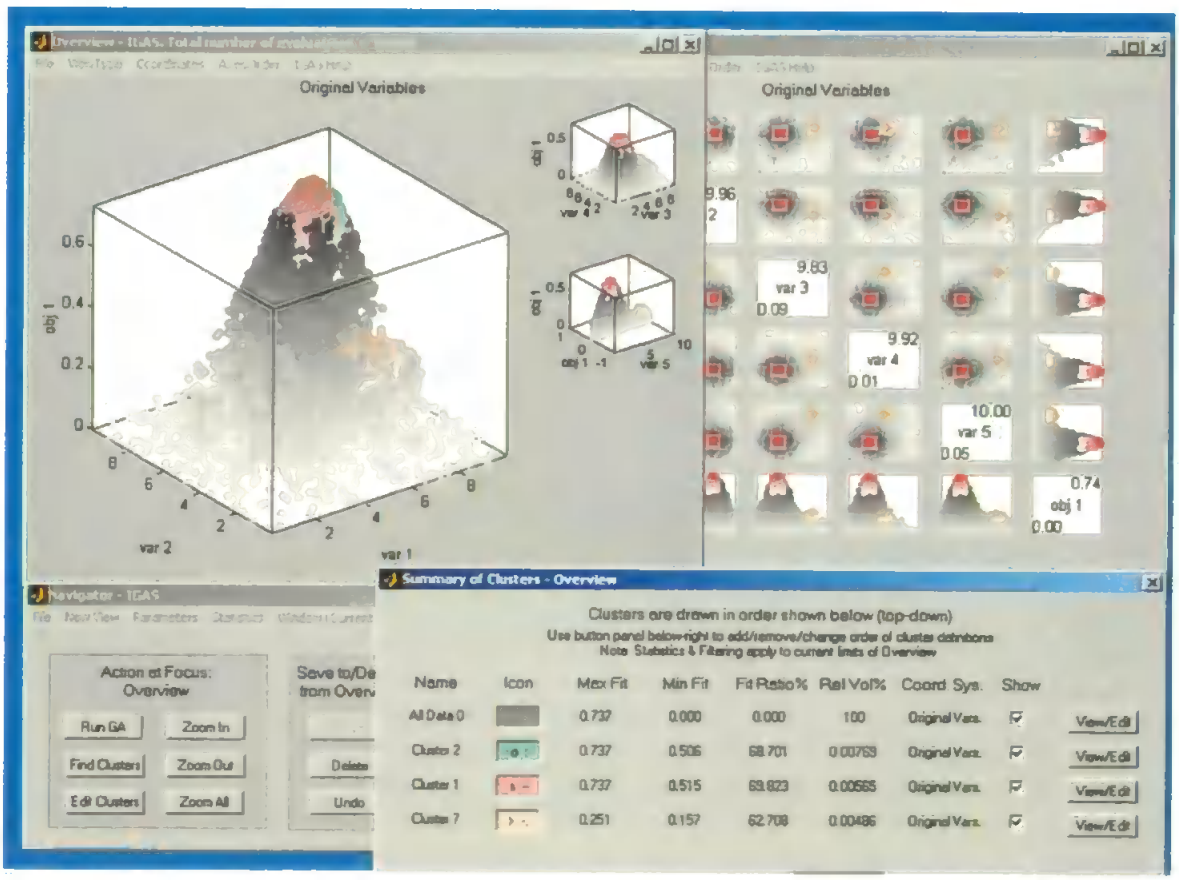


Figure 7.4: Example of User_4 final result (Test_1).

7.3 Summary of Questionnaire Results and User Comments

7.3.1 Empirical Results

A number of interesting findings were obtained from the formal feedback of the questionnaire. The users were asked to give a quality or preference measure by ranking the regions they have defined in the search space. The regions could be derived from the clustering algorithm and other tools such as the objective filter or defined manually by the user. Table 7.4 shows the raw results provided by the users in the questionnaires (see Appendix D), for each cluster identified by the user, the "peak number" of the closest ideal region (Table 6.2) is given here. To evaluate the accuracy of the "user rank" it is necessary to compare with the derived actual ranks, however these ranks were defined with full knowledge of the search space, while the participants only use the information they have found. Therefore the order of the ranks should be compared rather than the absolute value.

Function	User Name (No. Peaks Identified)	Peak Number	User Rank	Confidence in Rank	Certainty all Found	Usefulness of System
Test_1	User_1 (3)	2 | 2* | 3	4 | 5 | 2	5 | 5 | 5	1	5
	User_2 (2)	1 | 2	3 | 4	3 | 5	3	4
	User_3 (1)	2	4	4	3	4
	User_4 (3)	2 | 2* | 4	4 | 3 | 1	4 | 4 | 5	3	4
Test_2	User_1 (5)	1 | 1* | 2 | 2* | 2*	4 | 3 | 2 | 1 | 3	5 | 5 | 5 | 5 | 2	3	5
	User_2 (3)	2 | 1 | 3	4 | 2 | 2	2 | 3 | 4	5	4
	User_3 (3)	2 | 2* | 2*	3 | 2 | 2	3 | 2 | 2	2	3
	User_4 (2)	2 | 1	3 | 2	4 | 4	5	4
Test_3	User_1 (6)	16 | 15 | 14 | 12 | 13 | 15	5 | 2 | 5 | 4 | 3 | 4	5 | 5 | 5 | 5 | 5 | 5	4	5
	User_2 (4)	16 | 16* | 16* | 15	4 | 4 | 3 | 3	3 | 4 | 3 | 3	4	4
	User_4 (4)	16 | 15 | 14 | 13	4 | 3 | 3 | 2	4 | 4 | 4 | 2	4	4

*=duplicate

Table 7.4: Questionnaire feedback results (see Appendix D). For each identified regions the closest actual Peak Number (see Table 6.2) is shown and duplicate peaks are highlighted. The Rank provided by the users for each region and Confidence in that rank is given. Ratings for the Certainty that all regions have been found and the Usefulness of the system are also shown. For all ranks *1 is low* and *5 is high*.

Column 3 of Table 7.5 shows the actual quality rank of the peaks found by the participants (including duplicates), in the next column these ranks are normalised between the maximum and minimum values of the user rank (copied from Table 7.4). The sixth column then shows the difference between the normalised and user rank of peaks giving a fair comparison, the mean of the differences for each test is shown in the final column.

Function	User Name	Quality Rank of Peaks	Normalised Quality Rank	User Rank	Difference in Rank |Norm-User|	Mean Difference
Test_1	User_1	5 | 5* | 1	5 | 5 | 2	4 | 5 | 2	1 | 0 | 0	.333
	User_2	2 | 5	3 | 4	3 | 4	0 | 0	0
	User_3	5	4	4	0	0
	User_4	5 | 5* | 3	4 | 4 | 1	4 | 3 | 1	0 | 1 | 0	.333
Test_2	User_1	1 | 1* | 5 | 5* | 5*	1 | 1 | 4 | 4 | 4	4 | 3 | 2 | 1 | 3	3 | 2 | 2 | 3 | 1	2.2
	User_2	5 | 1 | 2	4 | 2 | 2.5	4 | 2 | 2	0 | 0 | .5	.167
	User_3	5 | 5* | 5*	3 | 3 | 3	3 | 2 | 2	0 | 1 | 1	.667
	User_4	5 | 1	3 | 2	3 | 2	0 | 0	0
Test_3	User_1	2 | 3 | 3 | 3 | 3 | 3	2 | 5 | 5 | 5 | 5 | 5	5 | 2 | 5 | 4 | 3 | 4	3 | 3 | 0 | 1 | 2 | 1	1.667
	User_2	2 | 2* | 2* | 3	3 | 3 | 3 | 4	4 | 4 | 3 | 3	1 | 1 | 0 | 1	.75
	User_4	2 | 3 | 3 | 3	2 | 4 | 4 | 4	4 | 3 | 3 | 2	2 | 1 | 1 | 2	1.5

*=duplicate

Table 7.5: Actual Quality Rank of identified peaks are given (see Table 6.2) and normalised to compare with User Rank (Table 7.4). The absolute difference between these ranks is used to evaluate the disagreement between User Rank and actual quality.

Table 7.5 indicates that the difference between the order of user rank and actual quality rank is generally low for Tests 1 and 2. The outlier is User_1 who ranked almost all regions incorrectly for Test_2 because of conservative behaviour; this user focussed on small regions of each peak, so receiving an untrue perception of the problem. The mean differences for Test_3 are a lot higher, this is because the quality rank was very different to that perceived by the users; the height of the peaks take on more importance for the users than the width (or hypervolume). The trade-off between height and hypervolume is

problem-dependent, it may be the case that the absolute quality of the product is more important than ensuring a robust design, therefore the users could argue that they were basing their conclusions on the purely visual information rather than the absolute mathematical value, which was all they had to go on in the absence of any more substantial design guidelines.

In many cases the same peak was identified a number of times by the same user as separate regions. Again the users are not necessarily wrong in making such a judgement, it is often difficult to tell whether two close regions are separate or part of the same region especially if they are at different fitness levels. Most "confidence in rank" values are over 3 (Table 7.4), so most people were conservatively happy with their answers. Some peaks received a "confidence in rank" of 2, these were usually peaks that the user did not have time or inclination to evaluate properly. In contrast User_1 seemed very sure of the rating but unfortunately was not correct. Nevertheless there is no overall correlation between "difference in rank" (Table 7.5) and "confidence in rank" for this small sample.

It is also instructive to examine the "certainty all found" and "usefulness of system" measures (Table 7.4). Users were least certain they had found all the important regions for Test_1, possibly because it was the first test they were presented with but it is also more difficult than the others to solve. Test_3 received a certainty value of 4 from all users when, in fact, they had not identified many of the peaks. This indicates that users thought they understood the search space better than they did; they ignored some of the desired regions (defined in Table 6.2) because the solutions were of low fitness and regarded as noise. Again a better explanation or definition of engineering design may improve this, but the psychological effects of the visualisation on decision making are apparent. Most users gave the system a high usefulness value (4 or 5) and were impressed with the tools provided for solving the problems. User_3 struggled to find any more information on

Test_2 and gave the system a 3 for usefulness, this user had the least experience and understanding of engineering design and evolutionary algorithms.

7.3.2 User Feedback and Comments

User_1

During the tests User_1 was very much in favour of using the scatterplot matrix where all 2D graphs are displayed at once and therefore never changed the order of axes shown. This participant used the clustering tool to find good regions and then attempted to find the correct tolerance using the objective filter mechanism at increasingly more stringent rates. The user rarely changed the clustering definitions supplied by the system and performed detailed analysis of those regions using a number of negative GAs. This resulted in a number of small regions being defined on the same peak and not much time was left for exploratory search (Figure 7.1). This user used the statistics on the summary of clusters dialog box to differentiate between regions as well as the visual interpretation on the screen. The behaviour of this user could be described as conservative and unquestioning of the clusters produced by the system, resulting in low success in terms of the statistical measures. However the participant enjoyed the tasks and gave a lot of constructive feedback. In subsequent discussions User_1 referred to the tolerance definition as an engineering 'constraint' and was more concerned with ensuring the feasibility of solutions found rather than maximising the size of the robust regions.

Direct feedback on the questionnaire from User_1 can be found in Appendix D.1. The comments given in Test_1 relate to a feature of the system that automatically redefines the fitness of regions when new data is added to the Overview Window (see Section 5.6 and Figure 5.18). This mechanism was confusing at first but then understood on the second test, other users mirrored this behaviour so a better explanation for this feature is required. User_1 also suggested automatically expanding the definitions of regions until they are

acceptable by the tolerance value (the opposite of the filter supplied in the interface), which is a good idea as long as users realise that, these new definitions are not the 'definite' answers (more data will be needed to confirm). The user also confirmed preference for the scatterplot matrix in the feedback for Test_3 and suggested finding the centre of regions to ease the zooming mechanism, however the definition of 'centre' is an ambiguous term – some users may consider the global maximum as the centre, while others the centre of mass in variable space and still others the midpoint of the ranges.

User_2

User_2 spent little time on the example functions but soon started on Test_1 and learnt how to use the system using this function, spending the longest amount of time out of all the experiments run. After investigating all the different available views User_2 settled on switching between the 2D Scatter and 2D+Fitness views. This participant used the 'Find Clusters' mechanism sparingly but then refined the clusters or created new ones manually using the 'Summary of Clusters' dialog (Figure 5.11) after some help was given. User_2 used the negative GA sparingly on Test_1, but afterwards preferred to make decisions and evaluate robustness using the visualisations rather than statistics. This user favoured the view types with the first two axes enlarged enabling detailed views of these axes, with the adverse result that little or no attention was paid to the other axes. Indeed on Tests 2 and 3 the 'AxesOrder' facility (Figure 5.9) was not used at all indicating that only dimension 1, 2 and fitness were used to make decisions and define the regions of interest (see Figure 7.2). The participant's behaviour can be describe as more exploratory resulting in a good success rate of finding new peaks and a good spread of solutions inside those peaks (high maximum peak ratio and inner hypervolume accuracy) but the dependence on visual information to evaluate robustness and lack of understanding of the tolerance and filter mechanism caused a low inner hypervolume accuracy and productivity ratio. User_2 gave values for the quality of regions found close to the required order (Table 7.5) indicating a

good understanding of the robustness criteria although the behaviour and online results indicate that defining the exact size of region was not considered important. This may be due to misunderstanding and lack of clarity of the initial instructions.

Formal feedback from User_2 was summarised on just one questionnaire (Appendix D.2). This participant noted that the data points were large on each plot meaning that very close solutions may appear as one solution and was concerned that points on the three-dimensional views were not given perspective; that is points 'further away' should be smaller. User_2 was the only participant to complain about this aspect of the interface, but an option to change the size of data points would be desirable, especially as the number of solutions increases. User_2 also asked for selectable zones in the three-dimensional plots to be implemented, however this is not a simple issue because the location of a cursor on the two-dimensional computer screen can mean an infinite number of positions on the three dimensional graphic. The participant also made a verbal comment that the zoom box on two-dimensional plots was sometimes difficult to see; it can 'disappear' behind lots of dark data.

User_3

User_3 performed just two evaluations on Tests 1 and 2. The participant used mainly the 2D Scatter and 2D+Fitness views, although the scatterplot matrix and 3D scatter were also briefly used. This participant used the clustering system to generate clusters, then refined them manually or by using the filtering mechanism. The negative GA was used to generate more data, then the user spent a long time editing and reviewing regions on the peaks already found. This is why three regions were identified for Test_2 that all relate to the same peak. More than one true peak was clearly visible, but the user preferred to define regions at the top of the tallest peak (Figure 7.3). The user was attempting to find a trade-off between the height and size of each region and used the statistic on the Summary of

Clusters to make decisions about the regions but sometimes got confused with the statistics and may not have understood the problem enough. User_3 fixated on the data generated in the first few GA runs and was not motivated to find more data by running a GA elsewhere. This participant liked to return to the Overview and did not change the axes order.

From the feedback on the questionnaires (Appendix D.3) it is clear that User_3 got lost when looking at different views, maybe because the data generated did not look like the Overview data, and saved the data to the Overview as soon as possible. The complaint about the volume on the Summary was due to a bug in the code that was rectified. This user enjoyed learning some parts of the interface and felt it was easy to familiarise with the system.

User_4

User_4 also used the 2D Scatter and 2D+Fitness views in the tests. After the initial learning period and explanation of the features this user exploited the negative GA and filter mechanism to assess the robustness of regions to good effect. The regions defined by the clustering algorithm were also used and edited although the user also defined regions manually in Test_2. This participant chose to ignore some regions and kept the useful ones after analysing them using the summary statistics and frequently changed the order of axes to check and define the clusters in each dimension. The user's behaviour was moderately exploratory discovering new peaks and analysing low fitness regions (Figure 7.4) resulting in a high productivity ratio, but the hypervolume statistic is poor because the regions were defined too small for Tests 1 and 2 and too high for Test_3. The conservative estimates were because the user was content to define the regions inside the tolerance, rather than to achieve the 50% tolerance exactly. Conversely in Test_3 the user became tired and finished the experiment without refining the size of the regions.

In the formal feedback (Appendix D.4) the user noted that the system crashed at the end of a session due to the bug in the statistical calculations. User_4 also said that the system was useful to solve these particular problems but some of the processes involved in the task could be grouped, such as a 'check robustness routine' that in the current implementation involves a certain set of button clicks. This would speed up the process and avoid the boredom problem. Verbally the participant suggested a number of modifications to the interface that could be implemented in a future version, these are summarised in Section 7.5.1.

7.3.3 Overall Summary

A number of features were not used by any of the users in these experiments. Viewing in alternative coordinate systems was demonstrated to the users but not immediately understood, so ignored in the tests. Similarly the parallel coordinate view was not used due to lack of familiarity with this representation of data. Very little changing of genetic algorithm and clustering parameters also took place, in some cases due to lack of understanding of the parameters. Those who had knowledge of genetic algorithms were dissuaded from changing parameters because of the limitation on number of evaluations allowed, they were content to allow the simple GA (with mutation scheme to avoid duplicated chromosomes) create new solutions if possible. The users were not encouraged to change parameters when they were introduced to the tests, as the evaluation was more concerned with visualisation and interaction aspects than the algorithms used.

The use of colours to label and edit clusters was a very accessible component of the system. The regions had to be defined by the users for results purposes but, after the initial difficulty of learning how to edit the clusters, the colouring system proved to be easy to understand and essential to "orientate" in the search space (Appendix D.3, Test_2). Unfortunately once a user has carefully refined clusters it is difficult to find new data by

'avoiding all highlighted clusters' as the GA will just find the lower part of the peaks already identified. Some mechanism to enable the easy duplication of clusters and to "expand the scope of every variable" (Appendix D.1, Test_2) is required to aid this process (although the amount of expansion will be crucial to further GA runs).

The participants would have struggled to learn all the different features of the system without the author there to guide them and answer questions. There was a variance in how much users understood the problem due to differences in their educational background and experience of relevant knowledge (see Table D.1 of Appendix D). However those with less experience were sometimes able to successfully interpret guidance from the author and tackle the problem as a visual puzzle to solve rather than as an engineering design task; this was confirmed in subsequent discussions with the participants. The 'puzzle' can be reduced to: locating regions, defining tight boundaries and trying to find a local fitness ratio of 50% by trial and error using the objective filter and negative GA mechanism. Some users tried to do the problem visually or with alternative strategies but quite often were not sure what to do next and needed prompting to 'look inside' a region or 'try outside'.

As reported, there were differences in how users interpreted the definition of quality and tolerance. User_1 was more concerned with 'beating the constraint' (tolerance) than 'meeting the constraint', others tried to meet the constraint while User_2 considered the constraint of little importance. All participants used the height of the peaks as the most important decision factor as the results on Test_3 testify, low fitness data (below 50% of the global maximum) was considered as noise by most users. Much of this 'noisy' data defined regions of the search space that satisfied the tolerance guideline (within 50% of a *local* optimum). The fact that many users were not aware of this indicates firstly that the system was not highlighting the relevant information, secondly that this definition of

robustness and quality was not well explained before the tests and thirdly the definition does not reflect how people generally interpret robustness and is possibly insufficient to be used on a general engineering design problem. All these issues are discussed further in Section 7.5.

Nevertheless the system facilitated the solving of this particular task, once the users understood it. The system also encouraged many different ways to solve the problems varying from purely visual to very analytical with varying results. The users studied here were very much novices on the system and the problem, although they were becoming comfortable with the system after a few hours use. This implies the system will be helpful when applied to a known problem, after the initial learning period.

7.4 Benchmark Algorithms Results

The benchmark algorithms: simple GA, sharing and deterministic crowding (DC) were run on the same three test functions. Table 7.6 and 7.7 shows the mean and standard deviation of the user experiment results (taken from Tables 7.2 and 7.3) compared with the mean of 10 experiments for each benchmark algorithm using 10 different random number seeds. These comparisons are made for interest only; they cannot be used to give a statistically confident analysis as more users (and algorithm runs) would be required.

Again the comparative difficulty of each test function and the characteristics of the peaks is reflected in the results; the algorithms generally agree that Test_1 is the most difficult to solve and Test_3 the easiest (although more difficult to understand than users realised, see Table 7.1). The results reveal the various strengths of the different paradigms, with no single algorithm dominating the others on all metrics. The users metrics are rarely the best, but also rarely the worst.

Function (No. Orig. Peaks)	Algorithm	No. Correct	Maximum Peak Ratio	Inner Hypervol. Accuracy	Product. Ratio	Combined Data Metric
Test_1 (4)	Users	2.5 (1.00)	.596 (.227)	.622 (.140)	.240 (.234)	.088 (.086)
	SGA	1.5 (.527)	.348 (.104)	.589 (.229)	.846 (.005)	.156 (.019)
	Sharing	3.2 (.632)	.773 (.133)	.752 (.123)	.160 (.063)	.090 (.033)
	DC	3.6 (.516)	.884 (.128)	.287 (.054)	.394 (.025)	.099 (.018)
Test_2 (3)	Users	3.0 (0.00)	.903 (.079)	.518 (.319)	.309 (.036)	.145 (.094)
	SGA	2.6 (.516)	.761 (.130)	.376 (.088)	.770 (.020)	.216 (.043)
	Sharing	3.0 (0.00)	.981 (.007)	.895 (.045)	.068 (.007)	.060 (.007)
	DC	3.0 (0.00)	.996 (.003)	.638 (.063)	.350 (.023)	.223 (.033)
Test_3 (16)	Users	15.7 (.577)	.952 (.047)	.611 (.244)	.633 (.109)	.381 (.187)
	SGA	15.3 (1.16)	.923 (.063)	.226 (.036)	.919 (.002)	.192 (.036)
	Sharing	16.0 (0.00)	.999 (.002)	.566 (.023)	.815 (.010)	.461 (.025)
	DC	15.5 (.850)	.923 (.049)	.266 (.048)	.705 (.024)	.173 (.030)

Table 7.6: Comparison of algorithms against user performance; mean (standard deviation). The mean results from Users 1 to 4 given in Table 7.2 are shown (not including User_0). For the algorithms the mean results from 10 runs are shown. The combined data metric is the product of the previous three metrics.

Function	Algorithm	Num Evals	Time (s)
Test_1	Users	26800 (6030)	3460 (1110)
	SGA	30100 (0)	106 (9)
	Sharing	30100 (0)	252 (12)
	DC	30100 (0)	87 (15)
Test_2	Users	23600 (9130)	2430 (592)
	SGA	30100 (0)	90 (9)
	Sharing	30100 (0)	228 (11)
	DC	30100 (0)	72 (16)
Test_3	Users	25200 (5560)	2240 (595)
	SGA	30100 (0)	93 (4)
	Sharing	30100 (0)	233 (25)
	DC	30100 (0)	69 (10)

Table 7.7: Mean (standard deviation) of time taken and number of evaluations.

In general DC had the best maximum peak ratio because once it finds a peak the algorithm performs local hill climbing to the top and does not lose the peak (in Test_3 however, the peaks are very close and it is possible that the small peaks are ignored once bigger peaks nearby are found, hence a lower maximum peak ratio). This local search means there are many duplicate solutions in the same place before the algorithm gets to the

top of the peak, therefore the desired region is sparsely populated causing a low inner hypervolume accuracy and productivity ratio. DC achieves the best overall data metric on Test_2; this objective function has peaks separated by a long way in the search space thus allowing each peak to be maintained in the whole population without disruption.

The sharing algorithm generally achieved good inner hypervolume accuracy because it is forced to find alternative solutions away from the good ones already found, so gradually extending to the edge of the required region but then extending beyond to the bottom of the peaks, thus reducing the productivity metric. Overall sharing performs best on Test_3 with a near perfect maximum peak ratio. This is because the peaks in Test_3 are evenly distributed throughout the search space and not separated by low fitness regions. Maximum peak ratio values for the other functions are less good because the search space is sparse and the few peaks are not evenly distributed, the sharing algorithm fails to find some of the peaks of Test_1 and is forced away from the local maxima of Test_2.

The simple GA is not very exploratory producing a low maximum peak ratio and generating a lot of solutions at the top of each peak causing low inner hypervolume accuracy but the best productivity ratio. In fact the convergence causes such high productivity ratio that the overall data metric is good for the simple GA on all test functions and the best on Test_1.

Compared with the GA results, the users achieved average results in all metrics with no particular metric good or bad; the standard deviation between users is high because of the small number of participants so it is impossible to draw any firm conclusions. Their exploration was better than the simple GA, as was hoped, but the maximum peak ratio for sharing (using domain knowledge) and crowding is generally higher than the users. They achieved good inner hypervolume accuracy on all functions because of the engineering

design task requirement, although were beaten by sharing on two of the functions. Productivity was less than average although this was not a characteristic they were asked to search for, indeed a productivity that is too high shows less exploration, as is the case for the simple GA. The users had the worst results compared to other algorithms on Test_1 because two of the peaks were very low and hard to find, possibly even ignored by some of the users. Their results were average on Test_2 although brought down by inner hypervolume accuracy because of the difficult in pinning down the size of the region in alternative coordinate system. The users achieved their best metrics on Test_3 even though there were a number of peaks not formally identified by them. The identified peaks were easy to define and lots of data was generated inside the regions improving hypervolume accuracy and productivity ratio. The mutation scheme (used by users only) also increases the diversity of data in local regions, potentially improving robustness.

Table 7.7 shows that the users used less evaluations in the experiments as they had the choice to finish when they wanted to although their time taken was a lot larger than the algorithms. This is to be expected as the users were thinking about the problem, comparing clusters and making informed decisions about regions whilst the algorithms did not. Of the algorithms, sharing was 2.5 times slower than the simple GA because of the similarity comparison, although crowding was faster than the simple GA because of the simpler selection function. The algorithm times could be greatly improved if they were run on a devoted machine in a low level language and are only given for comparison purposes.

In summary it is difficult to compare the user performance with the benchmark evolutionary algorithms because of the difference in goals and domain knowledge they have. Sharing and niche techniques are given knowledge that may not be present in real world engineering design problems. If the niche radius and location of optima are already known then the problem is already solved, however if some knowledge of the search space

is present (and patience to try different parameters), sharing can generate a number of diverse solutions. Without domain knowledge deterministic crowding is probably the better option to search for local and global optima, its reduced computational cost is also an advantage factor. The users had a specific goal that none of these algorithms were designed to do. The metrics give some idea of how the users performed, but with such a small sample it is impossible to make any conclusions. Even with a suitable number of testers and algorithm runs, a comparison may be impossible to undertake because of the differences described above. Therefore this quantitative analysis gave some instructive results but no real indication of the relative performance of the system; further work is required to assess how the system maybe assessed quantitatively. The qualitative analysis of the users feedback on the system is more valuable as it shows that the system has potential for solving the engineering design task and possibly encouraging creative design.

7.5 Critical Analysis

7.5.1 Interface Improvements

A number of interface improvements were suggested by the users and noticed by the author during testing. In particular the summary of clusters dialog is too big and contains a lot of functionality including listing the clusters as well as allowing editing and changes to the list. The 'Add Clusters' checkbox is also here, but should be in a more accessible place. User_4 suggested having a summary on each window so the user can quickly choose which clusters to view. Easy duplication and editing of identified clusters would aid and speed up the exploration and exploitation process, for example if a cluster definition can be easily expanded to cover a known peak in the search space, new peaks would more likely be found by the clustering procedure or a new genetic algorithm run.

If changes are made to the clusters then all windows should be updated immediately rather than waiting for a redraw in each window, such a feature would be

possible to implement if the system was written in a language that supported fast graphics. Recent coordinated view systems recommend that up to three multiple plots are given in a single window rather than a number of different, overlapping windows (Jern *et al.* 2003, Brodbeck & Girardin 2003); such an implementation may be ideal if the data is static, but may not be possible if various types of new data need to be compared with previously found data, as is the case after a new GA run. However most of the users preferred not to have too many windows open at the same time and closed them as soon as possible (after saving the data to the Overview Window if required). Therefore a review of the employment of windows and design of dialogs is required.

The system was not immediately accessible and some features were present that seemed unnatural when initially presented to the users. For example the system does not immediately zoom in on a region given by the mouse pointer but waits for an action that will cause a redraw, the 'Zoom In' button will do this, but also closing the 'Summary of Clusters' and 'Run GA' dialogs will have the same effect. This behaviour is convenient if multiple functionality is required, but was not immediately intuitive and may not be wanted in some scenarios. Features that were mentioned as badly designed by some users were not mentioned by others, for example the criticism by User_2 about the size of the points (Section 7.3.2) in three dimensional displays was not seen as a problem by the other users. Options to customise the behaviours described in this paragraph and other user-friendly details would need to be tested and included if the system is to be more widely used. Generally the users did become accustomed to the display and learned to use unusual features fairly quickly.

7.5.2 Reporting of Relevant Information

The first important outcome of the experiments is the difference between the users' definition of important or interesting regions in the search space and the ideal regions

given in Chapter 6, as the difference between "number of peaks identified" (Table 7.1) and "number of peaks found" (Table 7.2) testifies, particularly for Test_3. This can be partly explained by difficulties in understanding the task as is discussed further in the following sub-sections. However the discrepancy also indicates that information relevant to the task was either not noticed by the user or not brought to their attention. To solve this particular task the user needs to be aware of all local optima in the search space, even those of very low fitness, so that their robustness and quality can be evaluated. The clustering algorithm based on kernel density estimation (KDE) was designed to slowly reveal high fitness peaks for the user to consider (see conclusions to Chapter 4). In theory the tool will reveal all the peaks in a set of data if left to run indefinitely, for general data it is likely that low peaks will be at the end of a very long list. So the clustering tool in its present from was not sufficient to provide the required information. This could imply that more sophisticated tools are needed to highlight the relevant data, either more advanced clustering algorithms or intelligent ways of choosing between clusters that have been found.

It is likely that more sophisticated clustering algorithms, such as those described in Section 3.5, would return more accurate clusters in terms of this task if the parameters were set correctly. To set the parameters correctly or to provide more intelligent ways of choosing between clusters, either domain knowledge is required or a representation of users' preference needs to be incorporated into the clustering routine. For a given set of data, if the definition of the design task such as robustness or constraint requirements changes, then different clustering results may be required. Incorporating the preference information implicitly would be desirable, but this is difficult to implement and is the subject of other dedicated research (Noy & Schroeder 2001, see Section 3.6). Explicit representation of preferences is easier to implement and will return clusters closer to user specification (Levinson *et al.* 1979), although may require knowledge of specific clustering algorithms.

Therefore a more complete version of the current clustering tool is required (whether based on KDE or not) that allow users to input explicit preference information for specific tasks. Options should allow the user to search for clusters containing high density rather than high fitness, set bounds on the fitness or size of clusters to be returned (allowing only low fitness peaks to be returned, for example) and search for skewed or other non-uniform shapes of data. The current implementation of the clustering tool also allows cluster definitions to overlap each other, sometimes this is desirable but often the two clusters are describing virtually the same region of the search space; a mechanism to temporarily remove this behaviour should also be available. An even more flexible clustering tool will mean desired information will be brought to the users' attention and allow data to be interpreted correctly.

7.5.3 Experimental Set Up

One of the most prominent lessons of the user experiments was the length of time taken to complete the tasks; this was due to the number of system features the users needed to learn and their difficulty in understanding the engineering design task. All users needed help from the author to perform complicated operations and have certain features explained a number of times, suggesting it will take days to understand the whole system rather than hours. The author also provided hints on how to complete tasks, users interpreted the comments differently and some asked more questions and may have benefited from the help more than others. It was difficult for the author to be objective and give everyone the same advice as they were obviously struggling to assimilate the unfamiliar system and new task. These observations indicate that the tests were very ambitious, which was necessary to test as many features as possible with limited time and resources.

This preliminary evaluation was undertaken by subjects with limited amount of engineering design knowledge, although many of them have an educational background

connected with psychology (see Appendix D), so possessed at least a theoretical understanding of the cognitive process involved in problem solving and design. In discussions most of the participants confirmed that they considered the task as primarily visual problem solving. This reflects the conclusions of Tweedie *et al* (1996b), that visual tools can transform a difficult cognitive (or engineering design) task into a simpler perceptual task. However the obvious misunderstanding of the engineering design task (possibly due to interpretation of the initial instructions) made the problem even more difficult to grasp. The psychological literature acknowledges that users have difficulty solving ill-defined problems with a large search space; lack of knowledge and poor understanding of the problem definition compounds the difficulty (Eysenck & Keane 2000, p. 409). So these experiments could be interpreted as an observation of cognitive and perceptual problem solving, suggesting another use for the interactive system.

In theory Test_1 was the easiest problem to solve, as the peaks were all defined in the original coordinate system without interaction. In retrospect the users probably had the most difficulty with Test_1 because new peaks were difficult to find. In the other functions most of the relevant data could be seen in the first run of the GA, although the users did not appreciate that most of the clusters that could be highlighted and evaluated for robustness. This indicates again that the tests were over ambitious in trying to test too many features at the same time. Preferably this system would be tested a number of times by participants learning and testing individual features, building up knowledge of the system and problems, until more complicated tasks can be attempted. Tweedie *et al.* (1996b) confirm that the design of "interactive visualisation artifacts" is difficult and revealed that a number of changes to their design were required after intensive testing of the system.

It would be instructive to see if an expert working on their problem would form the same strategies as reported here. It is likely that the strategies used to solve this task by

novice users would be very different to experts solving there own problem, as experts encode the problem in a different way to novices (Eysenck & Keane 2000, pp. 393-426). Engineering design experts may have a very different idea of what robustness is and how to evaluate it. In addition different tools may be required to solve a particular engineering design task. Therefore this theoretical analysis can provide conclusions on the usability of the system in general, but their applicability can only be tested in real world scenarios.

Comparing the user results with the results of the unsupervised benchmark algorithm experiments was also very difficult to analyse as the different paradigms are all attempting to optimise or achieve very different goals. The multimodal evolutionary algorithms achieved results as good as or better than the users on some test functions, but this does not imply that interaction and visualisation should be discarded. Indeed the previous analysis implies that even if the perfect result was returned to the user, they would not realise it unless they knew exactly what they were looking for in the first place. The fact that the system encourages search and enables knowledge discovery, as seen in the users' results and comments, indicates the strength and potential of the tool.

7.5.4 Description of Engineering Design Task and Robustness

The description of the engineering design task was clearly difficult to understand. In future experiments a simpler task and clearer explanation is definitely required. Nevertheless feedback from User_4 confirmed that "some functions/processes could be automated or grouped in some way" (Appendix D.4, Test_1) indicating some appreciation of the task and the tools needed to solve it. Such a grouping can only be implemented once the sequence of steps has become known, so it is recommended that the low level features are kept but an advanced feature that allows the user to choose and edit multiple actions is made available. Two such prototype routines are described in Sections 7.6.1 and 7.6.2. Further analysis of the use of alternative coordinate systems is also given in Section 7.6.3.

As stated previously the engineering design task (in particular the definition of quality based on tolerances in objective space) was chosen to encourage exploration and decision making between regions of the search space by evaluating their relative fitness and robustness. The advice to evaluate regions in terms of local optima was misunderstood or ignored by some users; most concentrated on peaks whose maximum fitness was more than 50% of the global optimum. This may be a reasonable strategy for some engineering design problems, for example if there is an absolute minimum value of performance that should be achieved. However, in general the Taguchi design methodology would advocate minimising sensitivity to noise before changing tolerances to meet manufacturing specifications (Phadke 1989, pp. 33-34). In fact the description of quality given in the tests defines any region of the search space as 'robust' as long as the fitness of the solutions is within 50% of the local optimum. User_1 noticed this fact resulting in the definition of small regions as seen in Figure 7.1, these regions satisfy the tolerance constraint but would be combined or enlarged if the whole fitness landscape was known. It could be argued that this definition is too loose as an infinite number of regions could be defined for comparison at a later stage.

Instead of defining tolerance in relative terms in objective space, given as a percentage in these experiment, it may be more practical to provide absolute bounds for tolerance in terms of design parameters (variables) or manufacturing specifications. In this scenario regions of similar size in variable space would be defined and their minimum performance value compared with other regions; this suggestion is further elaborated in Section 7.6.4. This alternative evaluation of robustness may be more relevant to real world design situations where the limitations of specific materials and the absolute constraints on specifications or cost are known.

Again it is acknowledged that the evaluation of robustness and the perceived quality of a product is very much problem specific and may be impossible to express explicitly. However the original definition revealed insights into how people interpret robustness and assess quality as a trade-off between robustness and fitness. This analysis has also provided further understanding of the different definitions of robustness.

7.5.5 Future Work

The limited testing did show some learning and usability by the users that suggests optimism for the future potential of the system. How much the system enhances the search process is hard to evaluate and requires extensive testing with a number of users that are willing to learn the system over a long period of time, preferably working on a problem they are interested in. Once users (preferably engineers) have become familiar with the system they will then be able to perform the tasks using well-grounded skills and engineering knowledge, instead of just trying to solve the problem as a puzzle; the design task may be very different but the results should be much improved. This has been partially achieved in the case studies presented in Chapter 8; the system is shown to engineers working on their problem, admittedly the author was operating the system rather than the engineers themselves, but a lot of positive feedback was gained in this way.

For future tests it would also be desirable to measure the number of mistakes or identical runs a user (or algorithm) has made, to discover truly redundant and confused behaviour. Currently the user's actions are all saved to a data file, but these were only used in a limited way in this analysis. More extensive interaction information could easily be made available for analysis to evaluate specific cognitive behaviour (as in Convertino *et al.* 2003) and could be used to warn a user that they are about to generate duplicate information. It is possible that the system could use the actions of the user to build up knowledge of the current task (Moore *et al.* 1997) or make choices about future searches,

perhaps using a neural network like Takagi (1996). Even more ambitiously an attempt to capture the design process (Reffat & Gero 2000, Burge & Brown 2000) could be made by analysing the actions of the user to suggest how new designs were found.

7.6 Modifications Suggested by Evaluation Experiments

7.6.1 Check Robustness Procedure

Extensive investigations with the system by the author and discussions with the users suggested that an automatic procedure could be implemented to assess the robustness of regions. This process currently involves the following steps:

Check robustness procedure:

1. Choose a coloured cluster to be checked or define one using the zooming and 'edit clusters' facility
2. Optional: run a positive GA to ensure the best local optima has been found
3. Choose an amount to filter the region by using the Summary of Clusters within the tolerance guideline (but need to guess how much)
4. Run a negative GA on the redefined region (usually in a new window)
5. Save the data back to the Overview window, this will automatically redefine the region again when combining the new and old data.

REPEAT 2-5 until level of minimum fitness and fitness ratio is acceptable

All these actions could be made into one simple task (with a few parameters) reducing the repetitiveness of the procedure (suggested by User_4). If the resulting minimum fitness level is outside requirement the procedure can be repeated again with another parameter guess. This also becomes tedious, but it is difficult to pin down the exact filter level required because it depends on data that has not yet been generated. The other possible outcome is that the resulting minimum fitness is too high and the region will need

expanding to find a less conservative result. Such a process is difficult in the current system, in fact redefining the region is probably the simplest process, but should be made possible if required (as suggested by User_1).

7.6.2 Find Robust Regions Procedure

The procedure described in the previous section could be extended to letting the system search for robust regions as defined in the engineering design task automatically. Such a 'default methodology' can be easily implemented in the current version by continually searching inside and outside regions found. Such a methodology is similar to the sequential niche procedure of Beasley *et al.* (1993) where discovered peaks are "derated" by a sharing like fitness parameter allowing further peaks to be found. Similarly the system can choose regions to avoid using the clustering technique and avoid them using the built in 'death penalty' process (see Section 5.5). Such a procedure could be defined in the following steps:

Find robust regions procedure:

Start conditions: Number of Clusters to find N_R *and tolerance level f%*

End conditions: Maximum number of evaluations: e_{max}

1. Run GA with default parameters and limits
2. Find N_R new clusters (avoiding those already found)
3. Run Positive GA in clusters found – save to Overview Window
4. Find 1 cluster within each cluster found in step 2
5. Filter each new cluster to *f%*
6. Run negative GA in new clusters defined in step 4 – save to Overview
7. Run positive GA avoiding all clusters found - save to Overview

REPEAT 2-7 until run out of evaluations e_{max}

8. Delete clusters defined in step 2, keep clusters defined in step 4

The clustering procedure is used twice in each iteration – once in step 2 to discover broad clusters that the algorithm can avoid at a later stage and then in steps 4 and 5 these clusters are refined to assess the robustness of the region. The broad definitions are deleted at the end of the experiment and the refined clusters kept for measuring purposes.

The results[1] of such a procedure for two parameter settings on each test function is shown in Tables 7.8 and 7.9 and compared to the mean of the user results from Tables 7.1 and 7.2. In these experiments the 'user' metrics can be applied because the algorithm is attempting to define a robust region. The results of the user metric (Table 7.8) for the algorithms are not much better than those found by the users but the data metrics given in Table 7.9 are much improved (and beat most of the algorithm results of Table 7.5). The most improved results occur when the filter f is 20%, because more solutions fall inside the correct region so productivity and inner hypervolume accuracy is improved (see Figure 7.6), however the algorithm finds it easier to find more peaks when f is 40% (Figure 7.5). These results show the difficulty of setting parameters for general problems, in a similar way to the problems with the sequential niche procedure acknowledged by Beasley et al. (1993); the likelihood of finding new clusters from a current data set is very much dependent on the parameters put into the algorithm and in particular the size of the clusters defined by the system. If the cluster is too small, the next GA run 'avoiding other highlighted clusters' will find the bottom of a peak already identified, if it is too big, much of the search space will be ignored.

So a devoted 'find robust regions' algorithm can improve on the performance of the users and benchmark algorithms to some extent, although the improvement is still problem-dependent. Figures 7.5 and 7.6 shows this devoted algorithm still gets stuck and

[1] The results of these preliminary experiments, as in rest of the chapter, are included for interest only. A statistical analysis of many more trials would be required to make any firm conclusions.

generates too many regions on the same peak. The number of identified peaks is always 6 in these experiments because two new regions are found at every iteration of the procedure, but some heuristics to stop the algorithm if many regions are defined in the same place is required. A user using this methodology would hopefully correct this problem and delete unwanted clusters or merge them with others. The clustering algorithm or automatic procedure can be made increasingly more sophisticated, but at some point a user is needed to make decisions during or after the procedure to refine, delete clusters or start the clustering process in a completely new region.

Function (Num. Orig. Peaks)	Parameters	No. Peaks Identified | Correct	Correct Ratio	Fitness Accuracy	Hypervol. Accuracy	Combined User Metric
Test_1 (4)	N_R=2, f=40	6 | 3	.375	.855	.312	.100
	N_R=2, f=20	6 | 2	.167	.834	.196	.027
	Mean Users	*2.25 | 1.75*	*.354*	*.975*	*.207*	*.061*
Test_2 (3)	N_R=2, f=40	6 | 2	.222	.962	.455	.097
	N_R=2, f=20	6 | 2	.222	.899	.267	.053
	Mean Users	*3.25 | 2.00*	*.511*	*.914*	*.220*	*.079*
Test_3 (16)	N_R=2, f=40	6 | 3	.094	.943	.342	.030
	N_R=2, f=20	6 | 5	.260	1.00	.153	.040
	Mean Users	*4.67 | 3.67*	*.191*	*.963*	*.428*	*.064*

Table 7.8: User metrics for single results of 'Find Robust Regions Procedure'. 31500 evaluations used. Mean of Users' results (from Table 7.1) also given for comparison.

Function / (No. Orig. Peaks)	Algorithm	No. Correct	Maximum Peak Ratio	Inner Hypervol. Accuracy	Product. Ratio	Combined Data Metric
Test_1 (4)	N_R=2, f=40	4	.959	.699	.181	.121
	N_R=2, f=20	3	.735	.875	.350	.225
	Mean Users	*2.5*	*.596*	*.622*	*.240*	*.088*
Test_2 (3)	N_R=2, f=40	3	.984	.901	.218	.193
	N_R=2, f=20	3	.992	.833	.298	.246
	Mean Users	*3.0*	*.903*	*.518*	*.309*	*.145*
Test_3 (16)	N_R=2, f=40	16	.982	.753	.607	.449
	N_R=2, f=20	16	.986	.734	.802	.581
	Mean Users	*15.7*	*.952*	*.611*	*.633*	*.381*

Table 7.9: Data metrics for single results of 'Find Robust Regions Procedure'. 31500 evaluations used. Mean of Users' results (from Table 7.2) also given for comparison.

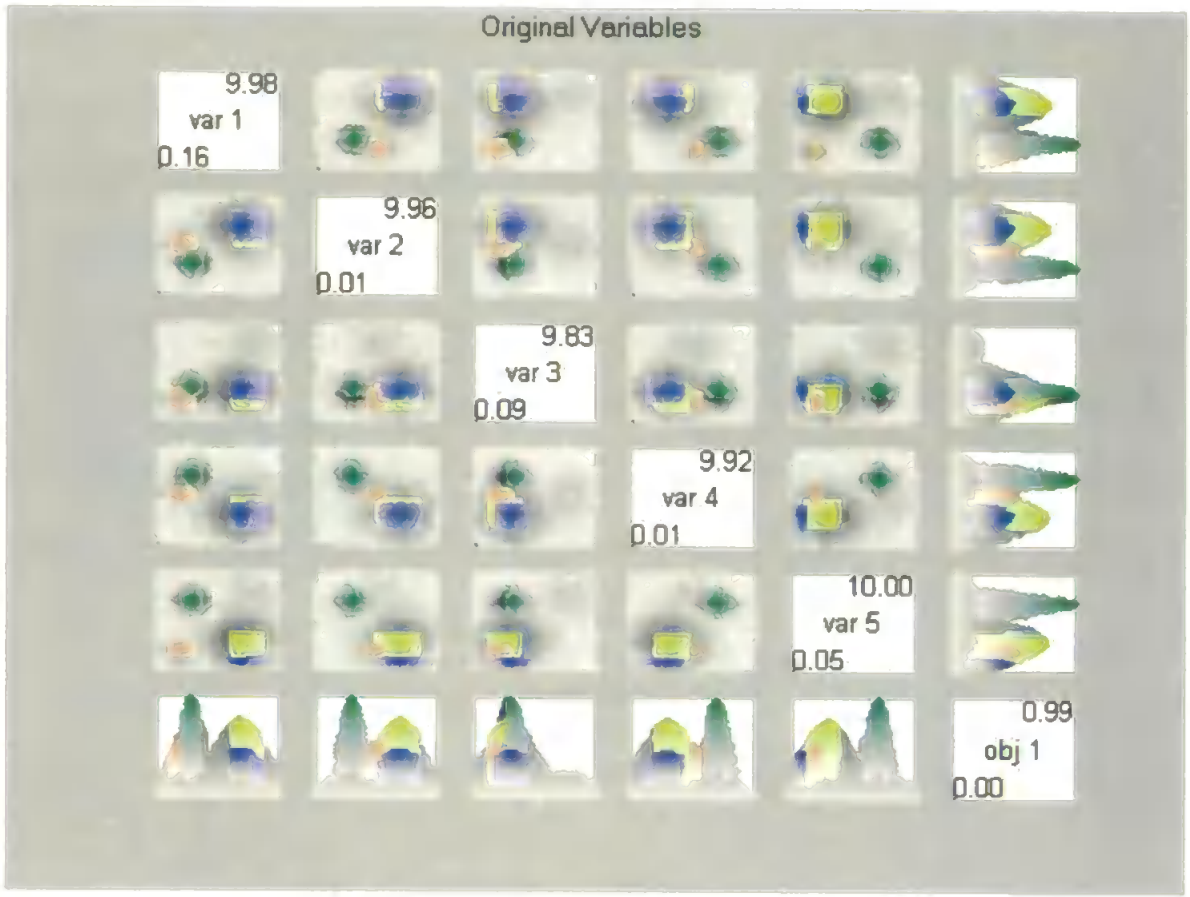


Figure 7.5: Test_1 regions found using N_R=2 and f=40%.

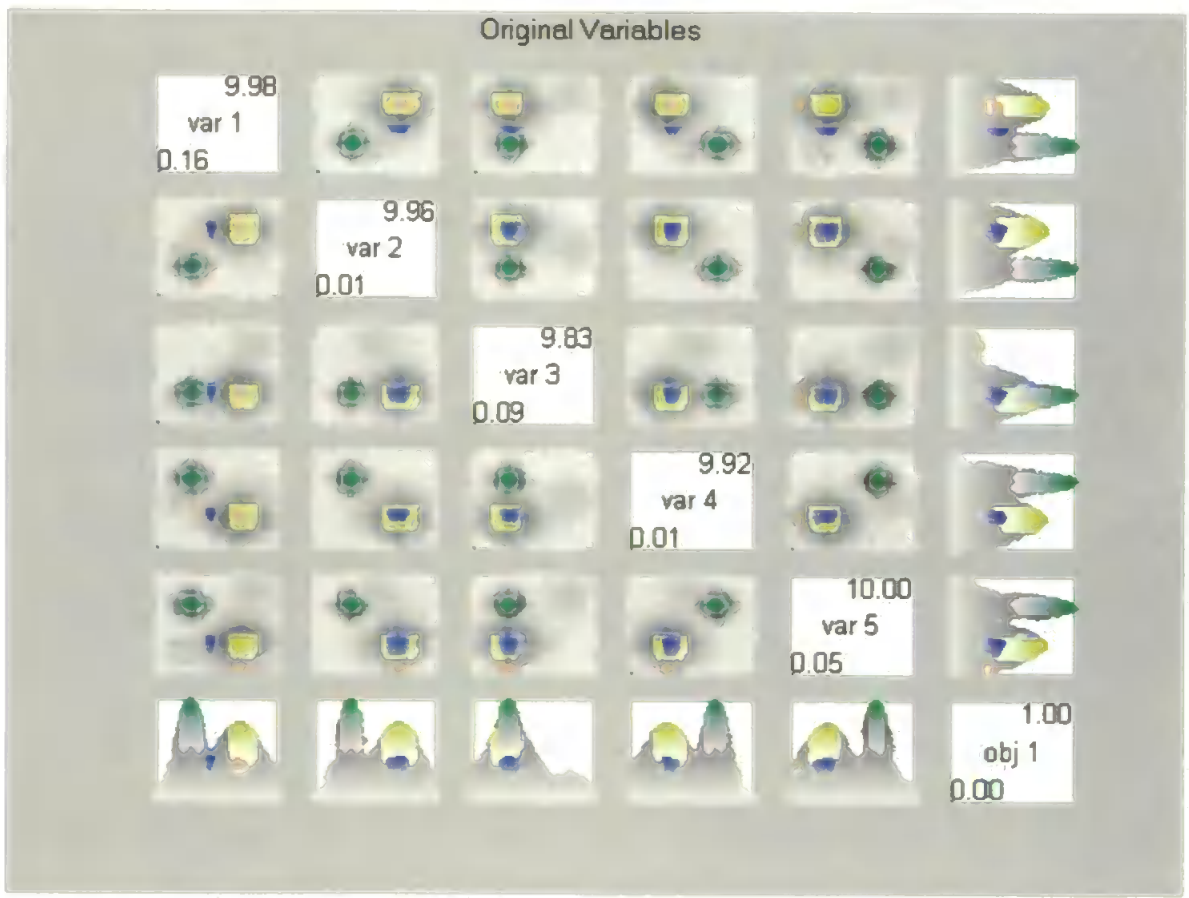


Figure 7.6: Test_1 regions found using N_R=2 and f=20%.

7.6.3 Defining Volume in Alternative Coordinate Systems

None of the users performed clustering in an alternative coordinate system. This is mainly because the concepts were difficult to appreciate on top of all the other features in the system. The question remains whether defining regions in an alternative coordinate system results in a closer definition to the true regions, particularly those regions defined naturally in another coordinate system. Test_2 contains two such regions so it is hypothesised that defining regions in the principal components and generating data in those regions will give a better fit. The 'Find Robust Regions Procedure' (Section 7.6.2) can be modified in the following way:

2a. Find principal component representation of N_R new clusters (additional step)

4. Find 1 cluster in the principal components of each cluster found in step 2

Tables 7.10 and 7.11 show the results[2] of a single run of the procedure on Test_2 using the same parameters as before (top two lines of each table). These results show no significant difference to the results from the experiments run in the original coordinate system (Tables 7.8 and 7.9). There is possibly a slight improvement for f=20%, at this level the cluster will be smaller and should define the true region more accurately, however further results are needed to prove the statistical significance of this. Another experiment is shown in Figure 7.7, illustrating search undertaken by the author on Test_2 using the principal components definition of regions at every opportunity, attempting to improve on the performance of Figure 6.4. The results are given as User_0_alt in Table 7.10 and 7.11, when compared with User_0 in Tables 7.1 and 7.2 there is an improvement on the user-defined statistics, particularly in the inner hypervolume accuracy metric, but the data metrics are almost the same.

---

[2] The results of these preliminary experiments, as in rest of the chapter, are included for interest only. A statistical analysis of many more trials would be required to make any firm conclusions.

There is also a question mark over the definition of 'relative volume' when applied to regions created in an alternative coordinate system. This value is found by transforming the overall data to the alternative system, then the hypervolume of the region in question is compared to the overall hypervolume. If regions are defined in a mixture of coordinate systems the relative hypervolumes may be incomparable.

From these limited experiments it is impossible to conclude whether defining regions in alternative coordinate systems is beneficial, although there may be some improvement on the user definition of regions. At the very least the alternative views can reveal new insight into the data and assorted clustering definitions that are sometimes useful. The technique is certainly interesting and deserves further investigation. More intensive experiments like those suggested in this section and analysis of the results will confirm the use and applicability of defining regions in alternative coordinate systems.

Function (Num. Orig. Peaks)	Parameters	No. Peaks Identified | Correct	Correct Ratio	Fitness Accuracy	Hypervol. Accuracy	Combined User Metric
Test_2 (3)	N_R=2, f=40	6 | 1	.056	.980	.213	.012
	N_R=2, f=20	6 | 2	.222	.904	.274	.055
	Mean Users	*3.25 | 2.00*	*.511*	*.914*	*.220*	*.079*
	User_0_alt	3 | 3	1.00	.999	.556	.555

Table 7.10: User metric statistics from single experiments searching in the principal components for 31500 evaluations. Compare with Table 7.8 (Test_2). User_0_alt is the statistics generated by the author using domain knowledge, shown in Figure 7.7.

Function / (No. Orig. Peaks)	Algorithm	No. Correct	Maximum Peak Ratio	Inner Hypervol. Accuracy	Product. Ratio	Combined Data Metric
Test_2 (3)	N_R=2, f=40	3	.968	.868	.134	.113
	N_R=2, f=20	3	.993	.953	.198	.188
	Mean Users	*3.0*	*.903*	*.518*	*.309*	*.145*
	User_0_alt	3	.999	.978	.254	.248

Table 7.11: Data metric statistics from single experiments searching in the principal components for 31500 evaluations. Compare with Table 7.9 (Test_2). User_0_alt is the statistics generated by the author using domain knowledge, shown in Figure 7.7.

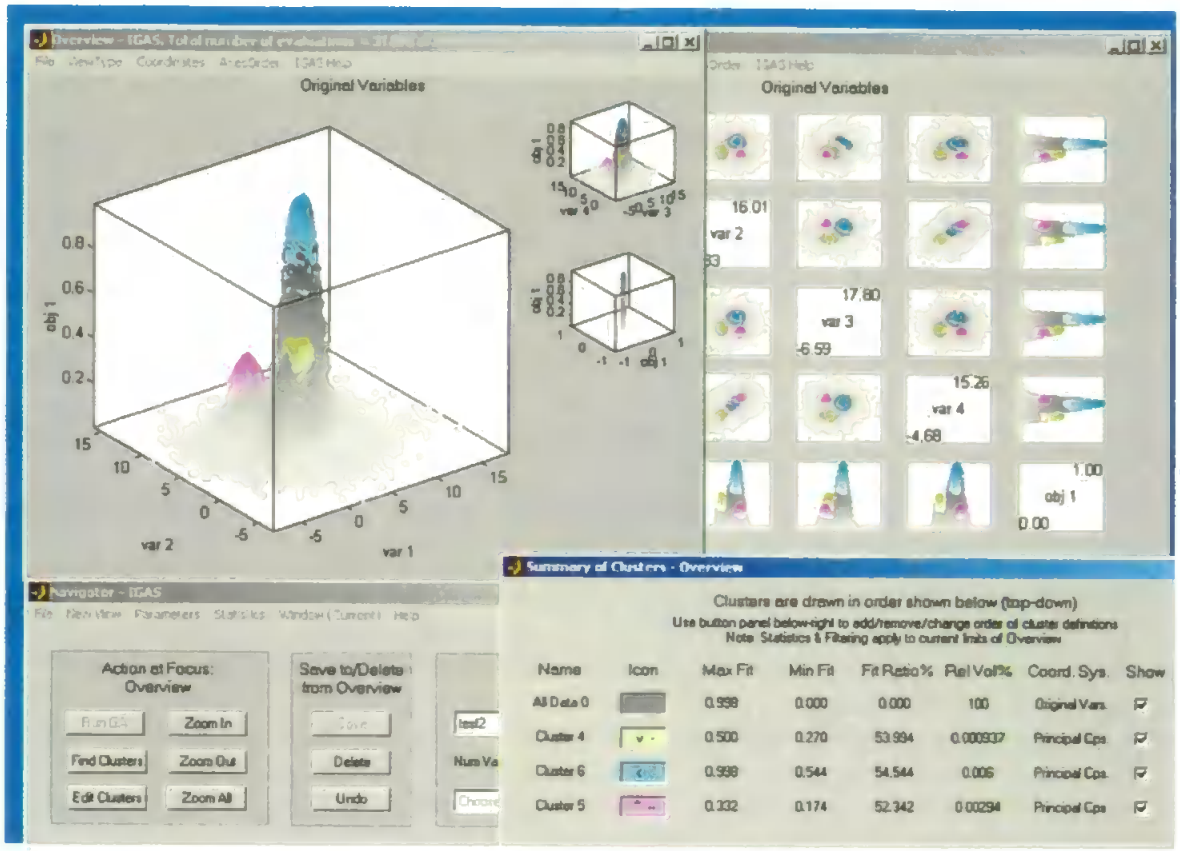


Figure 7.7: Regions of Test_2 defined in principal components, found using knowledge of the search space by the author. Compare with Figure 6.4.

7.6.4 Alternative Way to Define Robustness

The procedures described in previous sub-sections are relevant to the engineering design task in the user experiments but may not be suitable for certain objective functions or other engineering design tasks. The main problem with the definition of robustness used in the engineering design task is the need to find the tolerance in terms of a local optimum and compare minimum and maximum fitness locally. An alternative suggestion is to simulate absolute tolerances in variable space and compare the absolute values of minimum and maximum fitness between regions. This can be modelled by defining regions around interesting peaks with equal volume and performing a negative GA search to find the worst fitness in each region. Then the local maximum of each region can be compared with local minima of other regions.

Figure 7.8 shows the idea on Test_1. Each peak is identified by a cluster defined using very similar tolerances or variable widths. The green cluster (known as 'Cluster 2') has the best minimum fitness despite not containing the best overall maximum. This minimum fitness beats most of the other clusters' maximum fitness apart from the blue cluster ('Cluster 3'), which has a very low minimum fitness. The users all correctly gave the green peak a higher quality rank than the blue, despite having lower maximum fitness. But any ambiguity is removed by comparing minimum fitness of similar sized clusters instead of needing to evaluate a trade-off between size and fitness. A similar procedure is shown on Test_3 in Figure 7.9, with the result that almost all peaks have a very similar minimum fitness, so the quality of a region can be simply determined by maximum fitness alone. Using this definition the quality ranking returned by the users rather than those given in Chapter 6 are more appropriate. The results will change depending on the size of variable tolerance chosen, although wider tolerances are preferred to reduce manufacturing costs.

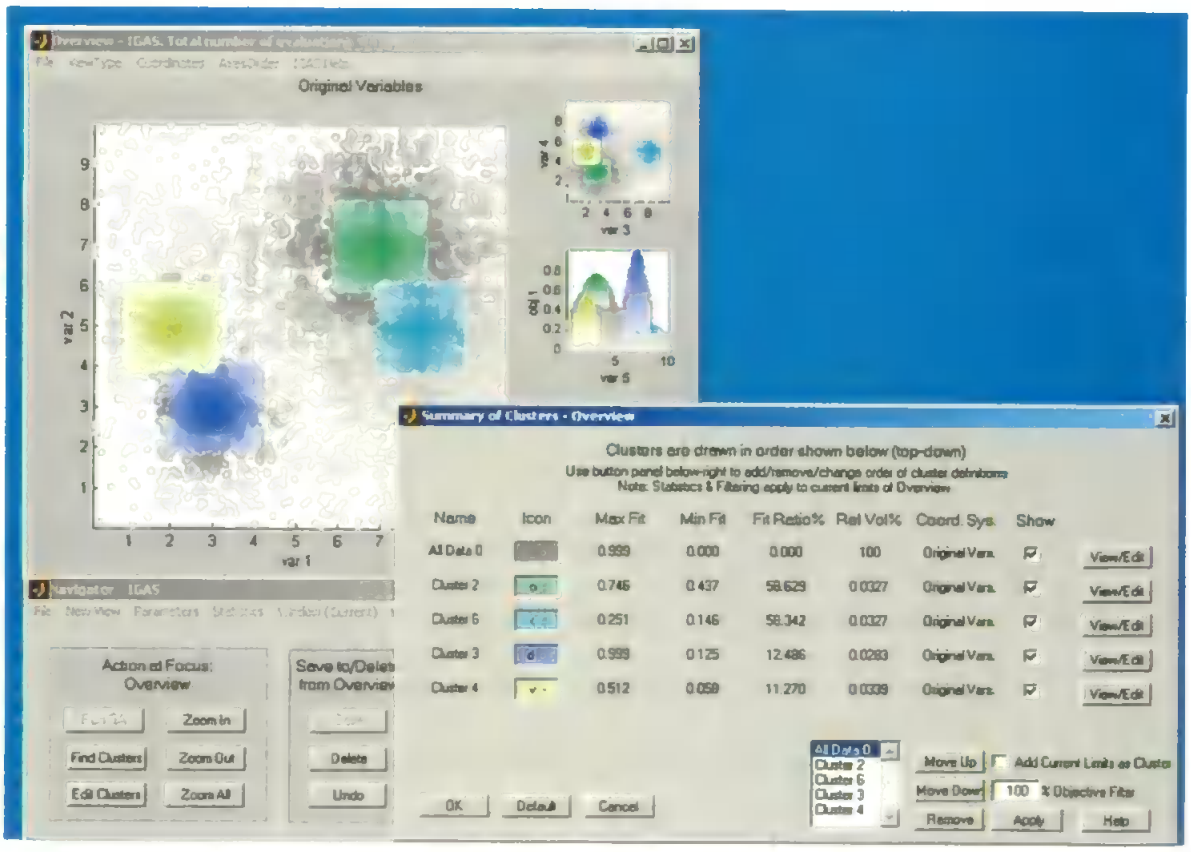


Figure 7.8: Alternative version of robustness: Test_1. Peaks are highlighted by clusters of similar size, negative GA used to evaluate robustness.

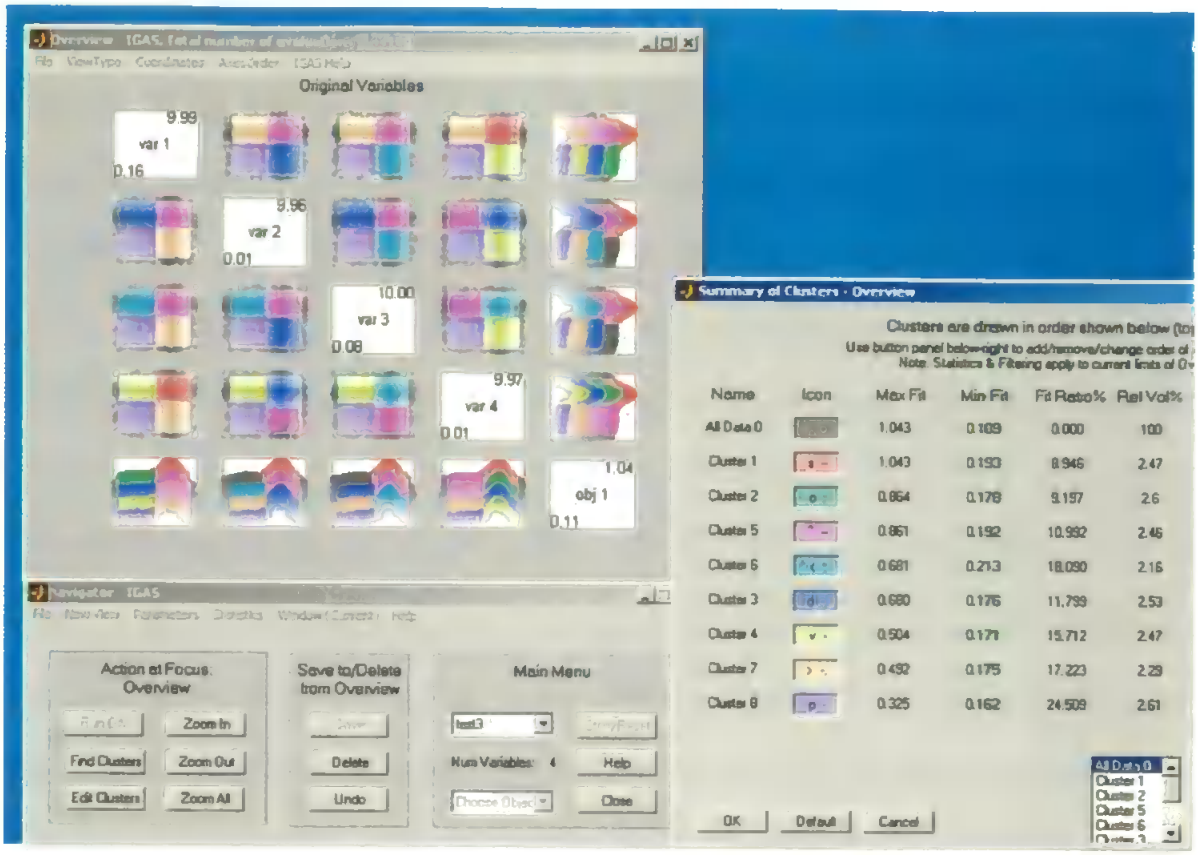


Figure 7.9: Alternative robustness definition: Test_3. The sizes of the clusters on peaks are almost identical; the minimum fitness of the clusters are also very similar. However if the volume of each cluster were reduced, the minimum fitness would reflect the order of the maximum fitness shown here.

7.7 Conclusions

The quantitative analysis was only partially useful due to the lack of users and time to conduct the experiments, statistically significant conclusions could not be drawn but some instructive results were found. The users were asked to complete an engineering design task; by manipulating the data and performing simple GA searches the users achieved some success. The test functions and metrics revealed differences in the behaviour of the benchmark algorithms and the users. The engineering design task could not be completed by any of the benchmark algorithms in its entirety; indeed it was shown that even a devoted hybrid algorithm would have difficulty solving the problem. These preliminary results imply that while a multimodal genetic algorithm will find data and information that is not directly relevant to a task, interaction with the simple genetic algorithm allows the user to search as desired and evaluate regions of the search space to their own

specifications. So the tests did show that the system achieved one of its requirements of allowing the user to guide the search and interact with the search space. However these tests were too ambitious to quantitatively show the advantages of interaction, suggestions on how to provide more reliable results are given later in these conclusions.

A reflective assessment of the users actions and feedback during the evaluation exercises provided a more reliable analysis of the system and tests than the quantitative results (Bucciarelli 1984, Schön 1984). The participants often enjoyed using the system and solving 'the puzzle' even if they did not fully understand the engineering design task. The visual displays helped them understand the problem; the many interactive features and different ways to view the data allowed them to get alternative perspectives on the data. The users were happy to generate new data and liked the clustering tool (sometimes preferring to define the clusters themselves) after the initial learning period. Individual comments and the interesting behaviour of users showed enough promise to continue.

A critical analysis of the user experiments taught some valuable lessons about the system, engineering design task and experimental set up itself. The system highlighted regions of high fitness and allowed the user to evaluate their robustness, but some information that was vital to solve the task was not apparent to the user and not made obvious by the system. According to the design task the users should not have ignored low fitness regions, so to some extent the system failed to achieve its other requirement of identifying potentially high performing regions. However the information required to solve this particular task may not be relevant to another engineering design tasks or different type of data. This suggests a more complete clustering tool is required that would allow the user to specify the range and type of data required. Better initial cluster definitions may be possible with more sophisticated techniques such as *k*-means clustering (that requires the computation of distances between data), Gaussian mixture models and projection pursuit.

Intensive study and comparisons between algorithms is required and could be undertaken in the future if the system, or algorithms, were written in a lower level programming language than MATLAB. Personal clustering preferences could then be incorporated into the system and analysis of the users' actions could be used to automate future searches and ultimately lead to further understanding of the engineering design processes.

Suggestions that some parts of the system could be automated were investigated in Section 7.6. It was shown that a procedure to check the robustness of regions and a default system methodology to find robust regions could be incorporated into the system. It has been shown that the filtering mechanism does work in alternative coordinate systems (Packham & Denham 2003). An experimental method to evaluate the accuracy of explicitly defining regions in this way was outlined in Section 7.6.3. If shown to be successful, the system would require testing by people familiar with the concepts.

The fact that the users failed to understand some aspects of the engineering design task prompted a review of the experimental set up itself. The task was designed to encourage decision making, but this led to uncertainty for the participants in deciding what actions to take and how to solve the task. The other main problem was the number of new features the users needed to assimilate even before attempting the problem. It was suggested that for future experiments, when more time is available, a number of users should be given the opportunity to learn and test each feature (see also Mumford 2003). Then they can be asked to solve explicit but increasingly more difficult tasks. Objective functions containing discrete variables, multiple objectives and discontinuous behaviour should also be included to more accurately simulate real engineering design problems.

The critical analysis of the engineering design task also improved understanding of the different definitions of robustness and use of tolerances. To evaluate the robustness of

regions in the search space it was necessary to find regions that satisfied a relative tolerance value set in objective space, then make some subjective decisions on the trade-off between maximum fitness and size or hypervolume of the region. An alternative way to evaluate robustness was suggested by applying fixed tolerances in the variables and then comparing the minimum and maximum fitness of the regions. This method allows direct comparison of robustness between regions and will be more applicable to engineering problems where limitations on materials and design parameters are known. The flexibility of the robust design methodology (Taguchi 1986) is retained as different regions of the search space can be evaluated for sensitivity and their objective values compared. This method is also similar to the setting of tolerances and analysis recommended by Tweedie *et al.* (1996) but is improved by the ability to check robustness using negative search.

In general there are many different ways of interpreting robustness and setting tolerances in engineering design (see Section 6.3). The system should be designed to allow tolerances defined in the design variables or in objectives. Both hard and soft constraints should also be supported so that engineers can evaluate whether designs are merely feasible or superior in terms of an objective value, whilst remaining robust. It is likely that each engineering problem needs to be solved with different tools and definitions, so a general engineering design system may be impossible to achieve, however the introduction of low level features and tools to set tolerances and constraint values will help.

These experiments involved users working on artificial test functions that they were not familiar with, they could be continued to help understand how people solve problems using cognitive analysis (Eysenck & Keane 2000). Although informative the procedure was very different to true engineering design behaviour (Norman 1986). The following chapter reports the comments and reaction of experienced engineers during the initial introduction of the system working on their design problems.

Chapter 8: Real World Case Studies

8.1 Introduction

This chapter describes the implementation of real world engineering design problems to validate the usefulness and effectiveness of the system; in particular to investigate the extent to which the interactive system helps engineers explore and understand the search space in real problems and whether new, robust solutions can be found. Such visualisation of the search space and the ability to compare the details of individual solutions is rarely available for practicing engineers. However real world engineering design problems bring a new set of challenges that need to be overcome. Some of the features in the system were less useful or needed modification to be applied in real world situations, in particular to multiobjective problems with conflicting criteria.

8.2 Goss Moor Rainfall Runoff Model

8.2.1 Description

This problem concerns the calibration of a model for the transformation of rainfall to stream flow. A number of procedures have been proposed to automatically calibrate such models (Sorooshian & Gupta 1995) and optimisation techniques have been used to find suitable parameter sets, for example genetic algorithms (Wang 1991) and genetic programming (Savic *et al.* 1999). The application of visualisation to this problem is particularly relevant as it has been acknowledged that many diverse parameter sets could return an equally good answer to the given problem (Beven & Binley 1992).

Mr. Martin Borthwick of the School of Engineering, University of Plymouth, developed the model given here from a time series approach given by Tsykin (1985). The aim is to find parameters for a conceptual model of river flow using the recent history of rainfall and previous day's river flow only, using the following equation:

$$R_i = R_{i-1}e^{-u} + vP_i^{w}P_{i-1}^{x}P_{i-2}^{y}P_{i-3}^{z} \qquad \textit{Equation 8.1}$$

where i = time (in days), e = exponential constant (base of the natural logarithm)

R = predicted daily stream flow (mm over catchment area)

P = recorded daily precipitation (mm)

u, v, w, x, y and z are dimensionless parameters (variables).

The units of the daily mean river flow (R) have been converted from discharge rate (m^3/s) to the equivalent daily runoff depth over the catchment area to be consistent with the rainfall units. A plot showing the measured daily rainfall on Goss Moor in Cornwall (UK) and the flow into the nearby River Fal is given in Figure 8.1. It is known that the river flow is modelled by more than just the previous few days' river flow and rainfall, other factors such as geology, vegetation and variation across the catchment should be taken into account, but a lot of the behaviour can be described by this model if the parameters are chosen carefully. A genetic algorithm (GA) can be used to find the best fit of the model by varying the parameters u, v, w, x, y, z and minimising the error between the predicted and actual river flow. The objective of the GA is thus to minimise the error between the predicted and observed river flow, given in terms of efficiency (r^2) according to the formula (Nash & Sutcliffe 1970):

$$r^2 = \frac{F_0^2 - F^2}{F_0^2} \qquad \textit{Equation 8.2}$$

s.t. $\quad F_0^2 = \sum_{i=1}^{N}\left(R_{obs} - \overline{R}_{obs}\right)^2 \quad$ and $\quad F^2 = \sum_{i=1}^{N}\left(R_{obs} - R_{pred}\right)^2$

where: F_0^2 = variance of the observed river flow

F^2 = describes the difference between predicted and observed river flow

R_{obs} = the observed river flow given daily (mm over catchment area)

R_{pred} = predicted daily river flow (mm over catchment area)

N = number of days (365), i = time (in days), r^2 = efficiency.

A perfect fit for this model would return an r^2 value of 1. Many of the recorded values for the precipitation data (P_i) were given as zero values, which will force the corresponding predicted values (R_i or R_{pred}) to be zero. To overcome this discrepancy zero values were replaced by very small values (10^{-16}), thus allowing a much smoother model to be formed. This procedure is acceptable because it is likely very small amounts of rain will have been recorded as zero. The three initial entries in the predicted flow were assumed to be the same as the first three observed values.

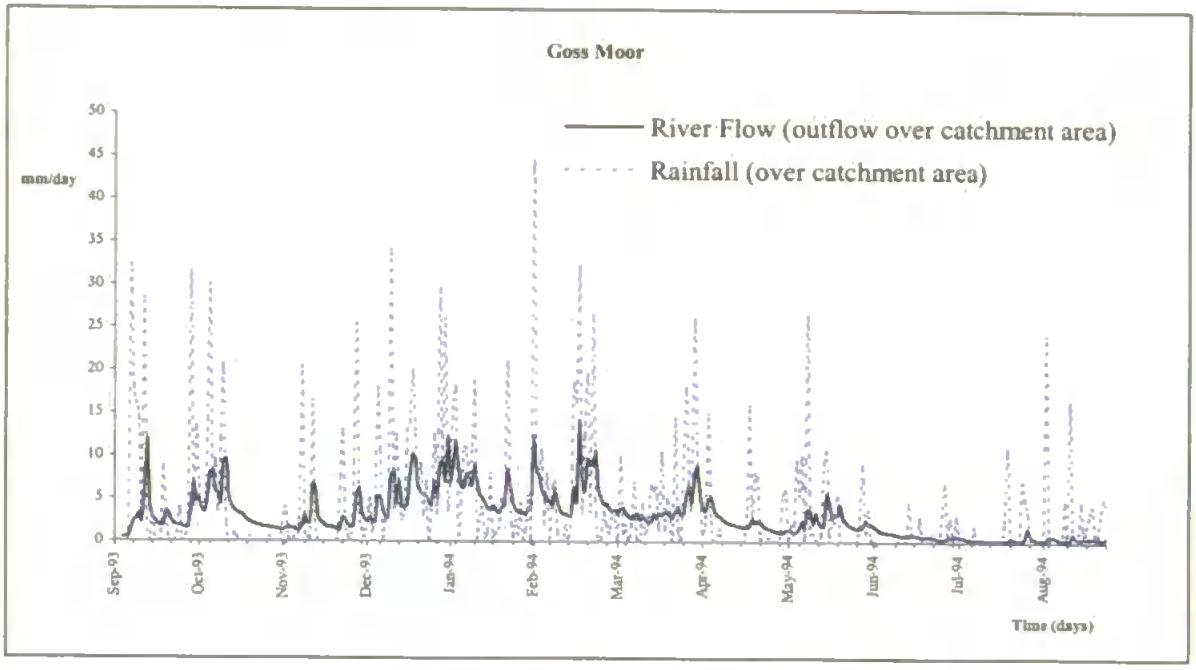


Figure 8.1: Goss Moor riverflow and rainfall data.

In practice, the results of such a process would usually be validated in order to check that a set of parameters holds for more than one set of data over the same catchment. Typically a long set of data is split into two, the first to calibrate (train) the model and the second to validate (test) it. In this study a single set of data (Figure 8.1) is presented and the model is used as a single objective optimisation problem with the intention of exploring the potential for visualisation and interaction in rainfall runoff modelling only. In future work it would be necessary to validate the results in some way (Davidson *et al.* 2000).

8.2.2 Expert Evaluation

The author of this thesis and the contributor of the model spent some time discussing the model and the results produced by the interactive visualisation system. The contributor already knew good parameter values and the limitations of the model due to his knowledge of the problem. However he found the system to be useful in confirming what he already knew and was impressed by the array of alternative results that could be accessed and visualised. Figure 8.2 shows an example of the initial data generated by the system using the default system parameters (running for 20 generations) with the variables u, v, w, x, y, z allowed to vary between 0 and 1. The best solution is shown giving an r^2 value of 0.663. Visualisation of individual results was implemented and shown to the contributor on the second time of meeting and noted as a significant aid towards understanding those results. Very small clusters were returned by the clustering algorithm indicating that the search space is very noisy but flat; there are a few very bad solutions with an r^2 of around -10^{-5} but also many diverse solutions in the more acceptable range $0 < r^2 < 1$.

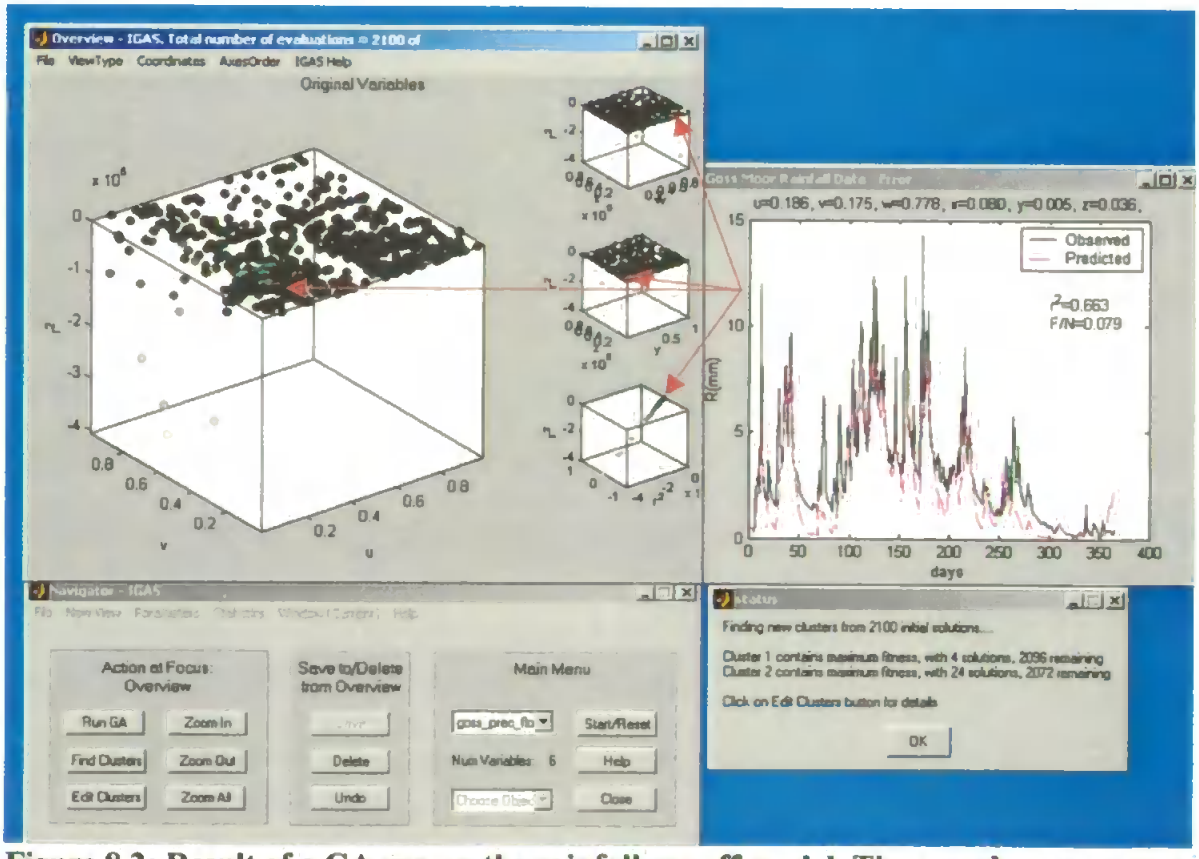


Figure 8.2: Result of a GA run on the rainfall runoff model. The search space appears flat except for very low fitness solutions. Individual details of the best found is shown.

The contributor used local search to find the best solution in the region suggested by the clustering algorithm. Running a GA inside the clusters was found to improve the error rate up to 0.768 (Figure 8.3). It can be seen that for both clusters found, x, y, z take very small values (0.1 or less). This suggests that the model should not include some of the older days rainfall as it has probably nearly all passed through on the same and possibly previous day. Therefore the last two terms (involving the y and z exponents) were removed from the model by manually changing the objective function.

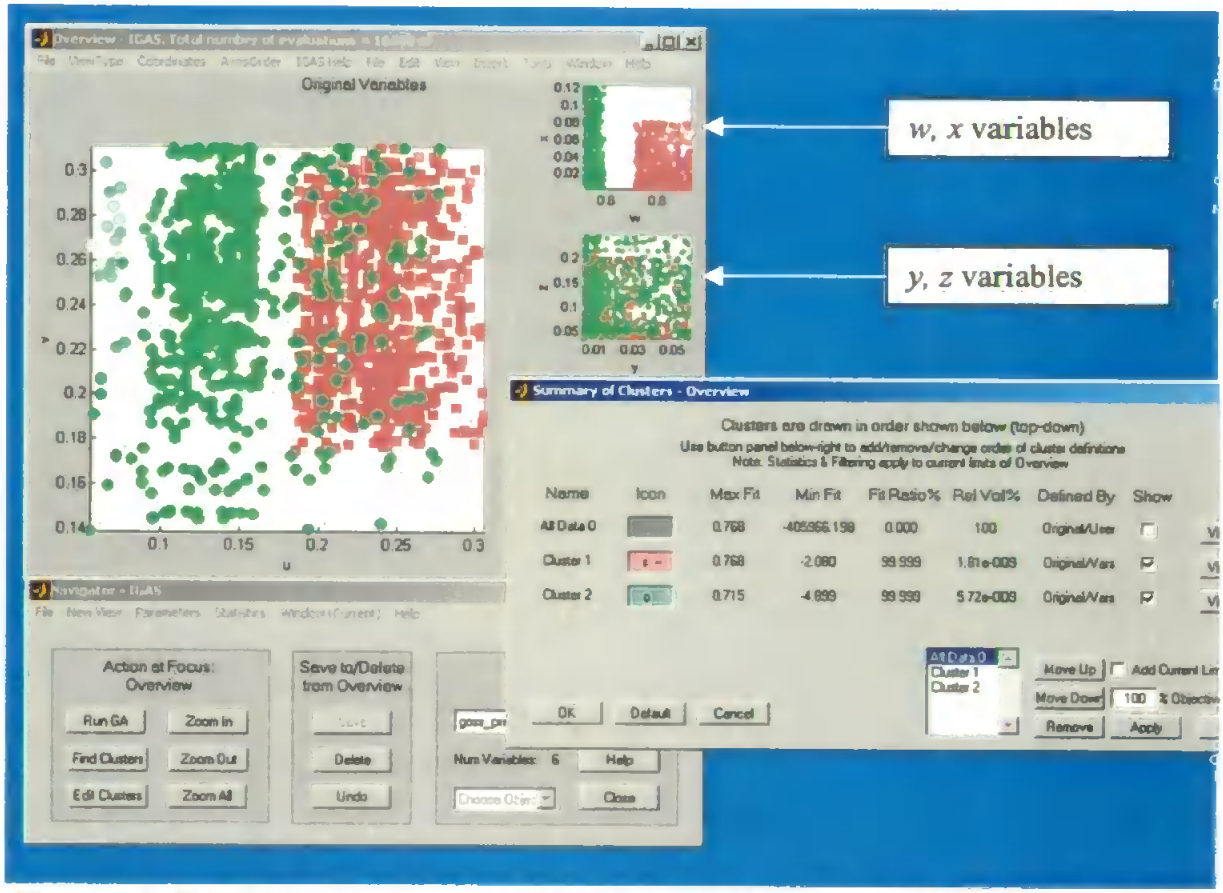


Figure 8.3: Local search suggests the high performance regions contain low values of x, y, z variables.

The result of a GA run on the revised model is shown in Figure 8.4. The search space is smaller so it should be easier to find an optimum solution. After two further local searches an efficiency of r^2=0.773 was found. These results suggest different regions give similarly good results or different parameters sets can give similar values of efficiency (0.773 and 0.772, Figure 8.4). The contributor used these results to seed a local search

method in Excel and achieved a result of r^2=0.803 with variable values u=0.153, v=0.105, w=1, x=0.020. This result is on the edge of the search space, particularly in w and x so may be difficult for the GA to find.

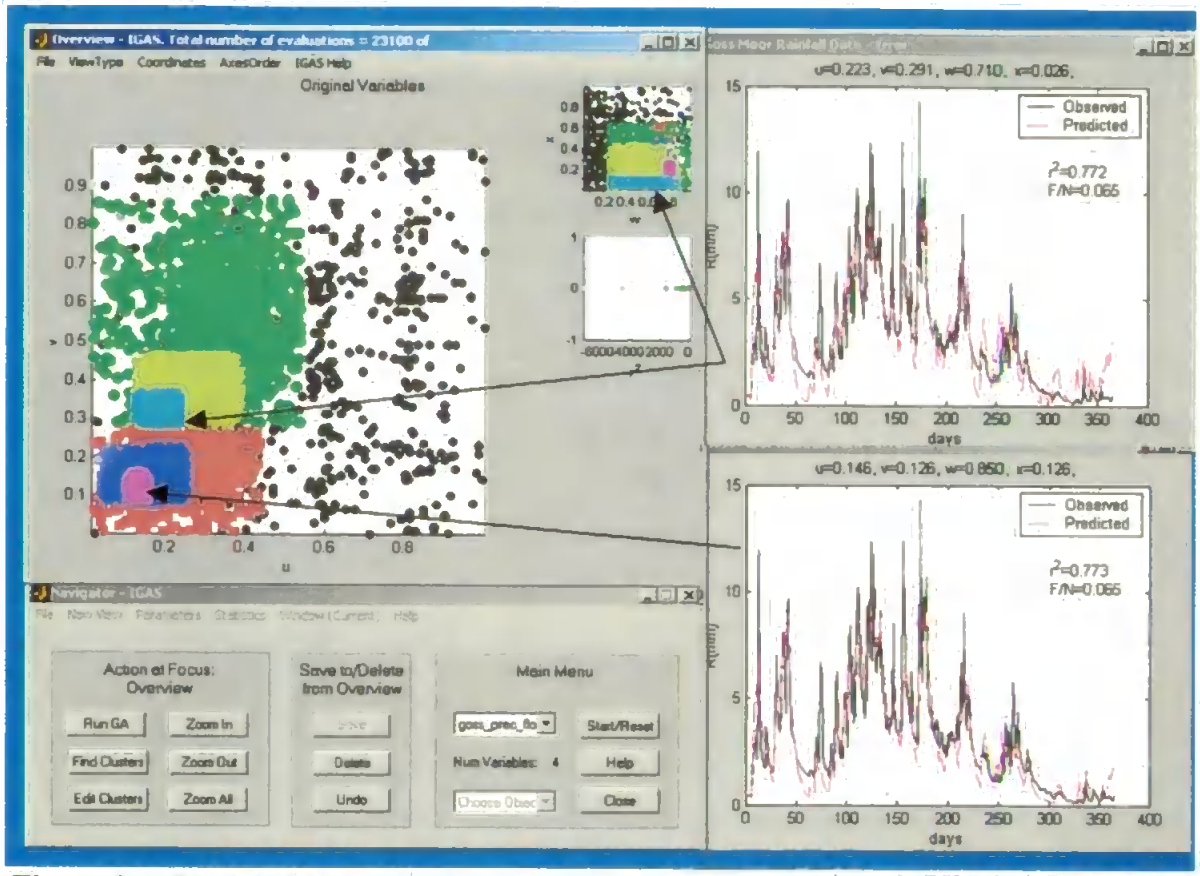


Figure 8.4: Model changed by removing unnecessary variables (in objective function) and run with four variables. After local searches in the two best regions, best error factors of 0.773 and 0.772 found.

A number of observations can be made about these initial results, mostly known to the contributor beforehand but confirmed by the system and visualisations. An efficiency of 0.8 is the best that will be achieved with this particular model. This implies that there are a number of factors missing in the model. A lack in the robustness of the model is suggested by the fact that many different parameter sets provide the same answer and each of the sets is quite sensitive to changes in parameter settings. In fact Figure 8.4 indicates that parameters u, v, x need to be constrained whilst parameter w could take on a variety of values and in fact may benefit from taking values larger than 1. Figure 8.5 shows the result of such a search and reveals a number of local optima in the search space. The best with an

efficiency of 0.801 has $w>1$, whilst the value of v has reduced to compensate (v and w both control the effect of the first day's precipitation in the model (Equation 8.1) and are therefore related). This is the best solution found so far, but how robust is it? The contributor used the visualisation to assess the robustness; the red region is small and isolated compared to the green, so is probably less robust. This example shows the amount of knowledge discovered from simple interactions and visualising a small number of figures is vast; without the visualisations such knowledge can only be obtained from painstaking analysis of results and interpretation by experienced engineers using their intuition.

The robustness of the regions can be confirmed using the tools on the interface. A positive GA was run inside each defined cluster (Figure 8.6), improving the best fitness inside the red cluster to 0.805, but showing some solutions in its vicinity are as low as –20, which is unacceptably low. To enable fair comparison the variable definitions of the red and green clusters were changed slightly so that they were about the same size (as suggested in Section 7.5.4 and 7.6.4). Strict filtering (less than 1% of the original) and negative GA search in both regions revealed that the red region contains worse solutions than the green region (Figure 8.7), thus the green is more robust than the red. Here the system has provided a piece of knowledge that could only be assumed by previous visualisations and would be very time consuming to confirm by iterative search.

The difference in robustness is confirmed in Figure 8.8, however the robustness of the green region is still questionable – a change of 0.02 in any parameter will result in a loss in efficiency. Figure 8.8 also shows the individual details of the best solutions. It can be clearly seen which parts of the data have not been well modelled by any of the solutions found, at around 50-80, 230, 270-300 and 340+ days (November, end of April, June and August) – possibly at times of unusual or changeable weather patterns.

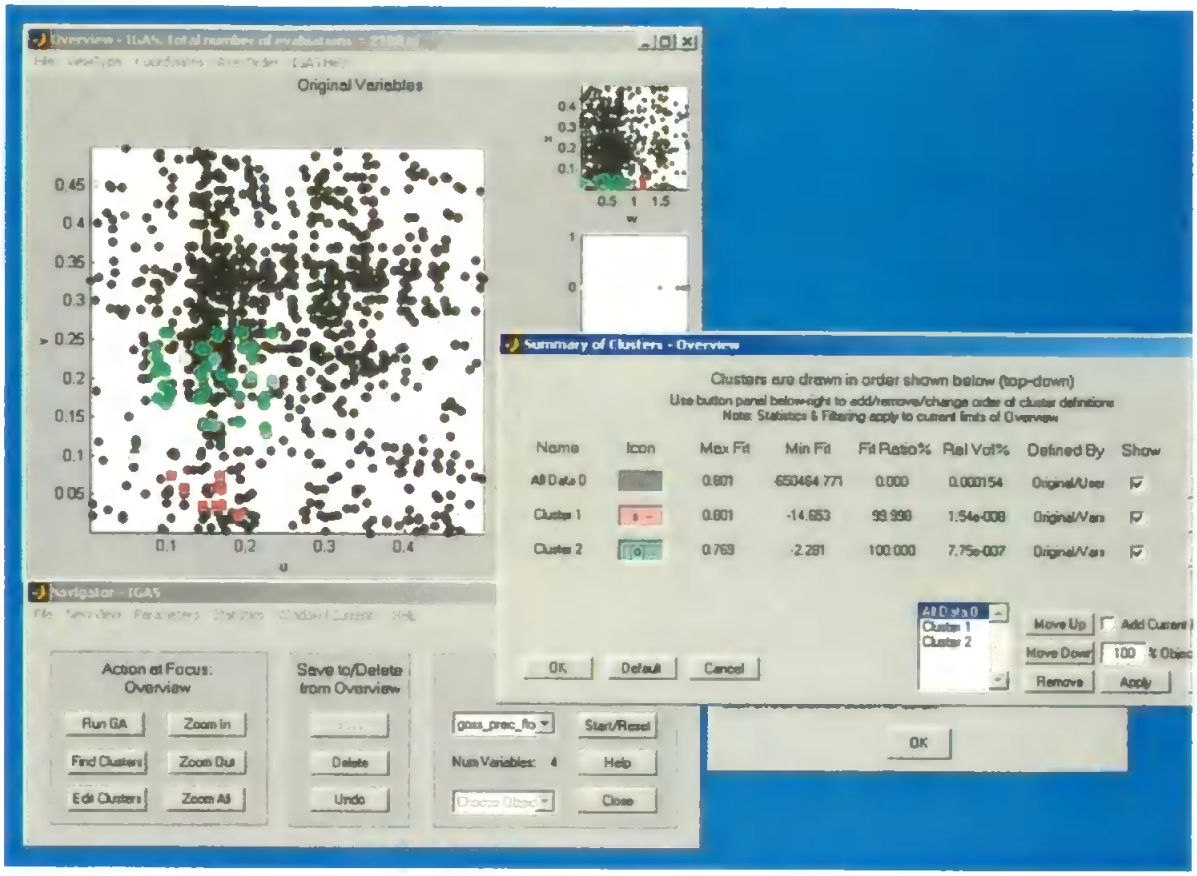


Figure 8.5: Variable limits changed due to analysis of Figure 8.4: $0<u,v,x<0.5$; $0<w<2$. Better solutions found around $w>1$, but how robust are those solutions?

Figure 8.6: Positive GA run in each high performance region improving the maximum fitness further, but some low fitness solutions also present.

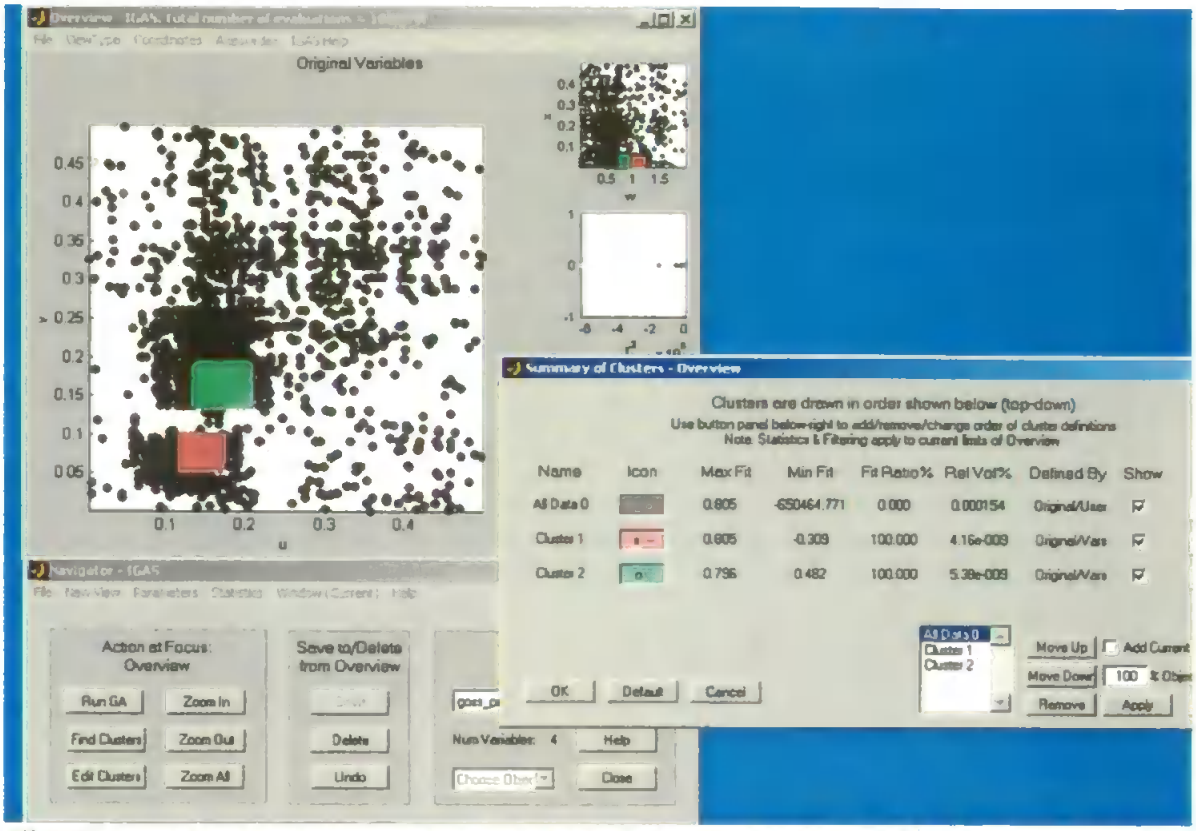


Figure 8.7: Check robustness by equalising volume of regions, filtering and running a negative GA. Analysis suggests the green region has better Min Fit so is more robust.

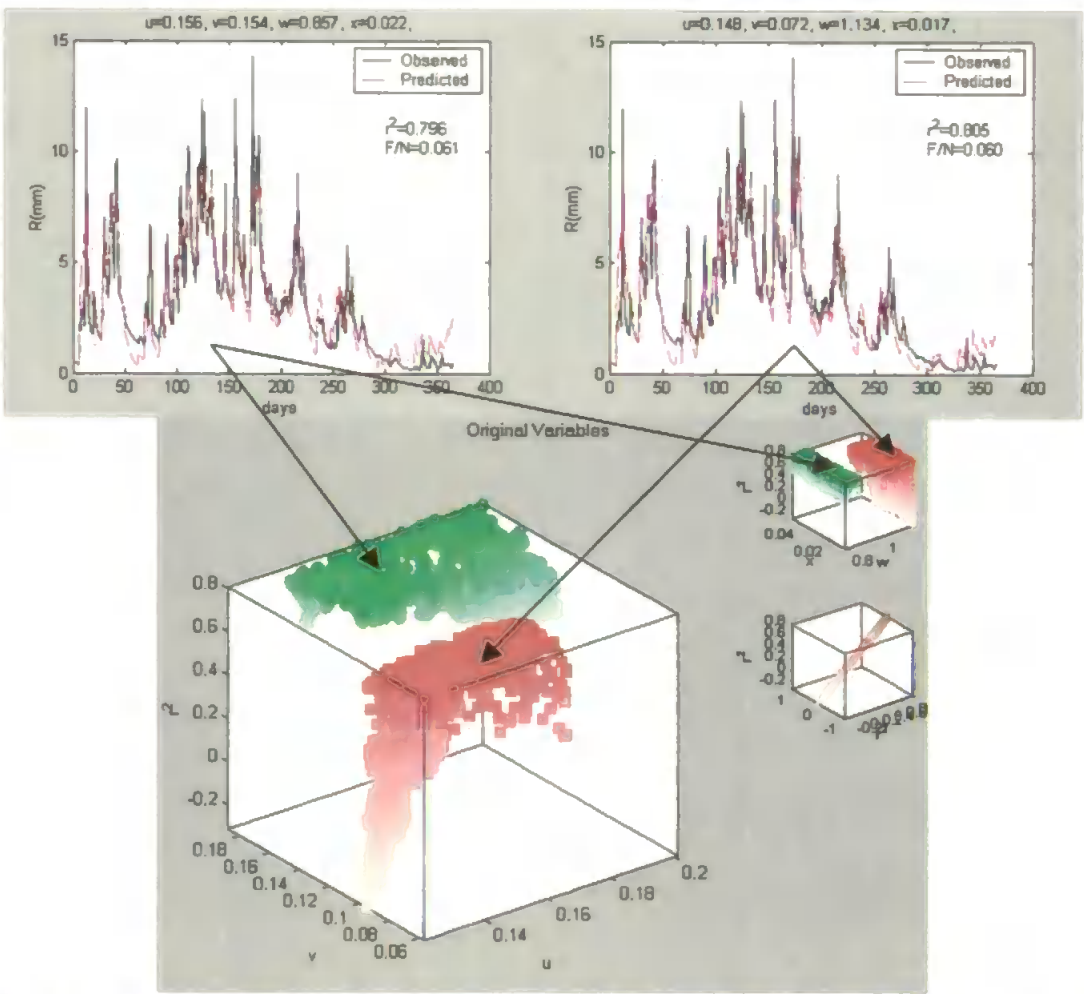


Figure 8.8: Two alternative solutions and regions with similar fitness, but the relative robustness is still not good; a change in variables by ~.02 will 'fall off' either peak.

8.2.3 Conclusions and Further Work Potential

Feedback from the contributor was positive in stating that the system helped to confirm results and understanding of the model (see questionnaire given in Appendix E.1). Being able to investigate interesting regions of the search space using visualisation tools within the system was particularly useful to assess alternative variable settings and then optimise using a local search method. The system helped to confirm that the later terms in the model were not necessary because rainfall from two or three days before has little affect on the river flow, verifying this was not possible without proper visualisation. This enabled a more detailed search of the reduced model (after manual modification of the objective function) with only four parameters. Feasible solutions were also found outside the original search limits, although this had the potential to change the model somewhat. The best solutions had very different values in those parameters that correspond to the first day's precipitation (v and w in Equation 8.1), revealing the amount of interaction between these parameters (see Figure 8.8). So creative design was supported by the system.

A number of different parameter settings can be found in the system that return similar fitness, but none improve on an efficiency of around 0.8, showing that the model is insufficient to fully describe river flow. The analysis could easily be further extended by reducing the number of parameters still further or by adding parameters to the objective function; for example in the exponential terms to model more precisely a certain day's rain. However the practitioner knows that geological factors such as the types of soil and vegetation that affect the drain rate of different parts of the catchment should be taken into account, as well as the effect of evaporation. Extended models such as those described in Davidson *et al.* (2000) could be quickly tried and tested using this system.

The visualisation of individual solutions was also noted as a significant improvement on the original display of just showing parameter settings, as they lead to a

better understanding of the search space and problem. The individual solutions show where the predicted data diverges from the observed data, suggesting that if seasonal changes and long term weather could be incorporated into the model the quality of the prediction would be improved. Alternatively the badly modelled parts of the data could be isolated and a different parameter set found using the current model, reducing the sensitivity of the landscape and presumably providing more distinct robust regions.

It was suggested that further work could be undertaken with the system. Because the objective function can be easily changed in MATLAB, it would be instructive to find the optimal number of parameters needed to model the data. However the model actually changes when parameters are added or removed (because different raw data is being used), but this is a valid form of testing the model; traditional engineering sensitivity checking involves fixing all parameters but one and optimising each one separately until the best is found. This is unnecessary when using the GA and within the system as described in the previous sub-section where wide initial limits were set and then interesting regions were 'homed in' on. Comparing the results of the two strategies would be instructive for this problem and general engineering design practice.

The system partially answered the robustness question by the visualisations and negative GA experiments. The conclusions were that changes in parameter settings will significantly affect some if not all solutions to the problem. From these results the lack of robustness of the model and sensitivity to variables would indicate that a parameter set from one year would not correctly predict the flow for another year. Normally the split records approach or another technique (Davidson *et al.* 2000) is used to determine the optimal number of parameters to achieve the most consistent results; this study shows that visualisation could be used to speed up these validation processes.

8.3 Biaxial Column Design 1

8.3.1 Overview of Biaxial Column Design Problem

Rafiq & Southcombe (1998) introduced an approach to find configurations of reinforcement bar placement in a biaxial column using a genetic algorithm. The goal is to place reinforcement steel bars efficiently in concrete so that the column will efficiently resist an applied axial load and bi-axial bending whilst minimising the amount of steel used. For some columns the bending moment is about a single axis (the uniaxial case) but for others the bending moment is about both axes (biaxial). This situation needs to be taken into account when designing the column. The British Codes of Practice (BS8110 1985) recommend a method of biaxial column design by simplifying it to approximate a uniaxial column. This method is only suitable for small columns and may lead to unsafe designs for larger columns where more reinforcement bars are needed.

To accurately predict whether a larger column will withstand an applied loading it is necessary to calculate the resisting capacity of both moments in a biaxial column for a given reinforcement bar configuration and check whether they satisfy the required conditions. Rafiq and Southcombe (1998) used a genetic algorithm to evolve solutions that satisfy the conditions whilst also minimising the amount of reinforced steel used. This problem is multiobjective with conflicting criteria (minimise the amount of steel and maximise the resisting capacity of the column). A number of designs will be produced by this method, but the engineer has a decision to make in the trade off between the two objectives. The system described in this thesis is an ideal tool to explore the search space for a variety of feasible solutions and visualise different configurations to compare their performance and suitability, but the discrete nature of the inputs and multiple outputs to the problem was such that not all of the features could be used and some modifications to the system were required to find better results. The objective function was kindly contributed for analysis by Dr. Yaqub Rafiq of the School of Engineering, University of Plymouth.

The inputs to the model are shown in Figure 8.9. They are the ultimate axial load N and the applied design moments M_x and M_y about the x and y axes that are to be achieved. For a given design the reinforcement bars will be positioned by the column detailer; to ensure "buildability" of the column there is a minimum spacing requirement for the bars so that concrete can easily be placed between them. The bars are also manufactured at predetermined diameters. These constraints were designed directly into the objective function with the result that the decision variables (x, y position and diameter d of each reinforcement bar) take on discrete values using the GA.

The outputs from the model, shown in Figure 8.10 are the maximum uniaxial moment capacities M_{ux} and M_{uy} of the section (concrete and bars) due to the ultimate axial load N. A method to determine whether the designed column will withstand the applied load, taking into account biaxial bending, was suggested by Bresler (1961). It was adopted by the British Codes of Practice (CP110 1972) and is based on satisfying the load contour equation:

$$L^C = \left(\frac{M_x}{M_{ux}}\right)^{\alpha} + \left(\frac{M_y}{M_{uy}}\right)^{\alpha} \leq 1 \quad \text{where} \qquad \alpha = \frac{2}{3} + \frac{5N}{3N_{uz}}. \qquad \textit{Equation 8.3}$$

Table 8.1 gives definitions of all symbols and values used in this equation and the following text. N_{uz} is the ultimate resistance to pure axial load; the maximum load the column will take before breaking in any direction, α takes ranges between 1 and 2, these values depend on the design of each column. When the load contour constraint is maximised ($L^C = 1$) an "adequate" approximation of the required relationship between the moments in the x and y direction is achieved (Beeby 1978), although the design will be conservative particularly when N/N_{ux} and the relative amount of steel used is low. N_{uz} and α are dependent on the design of the column, but are not optimised as objectives, so are known as 'dependent variables' in Table 8.1. M_{ux} and M_{uy} are the uniaxial capacities but to

be certain that the configuration will meet the design specifications the biaxial resistive bending capacities (MR_x and MR_y) should be greater than the design moments (M_x and M_y). In the absence of software to determine the resistive capacities exactly (as is the case in this study) Rafiq & Southcombe (1998) state that the load contour method is a satisfactory method for checking the validity of a given column design.

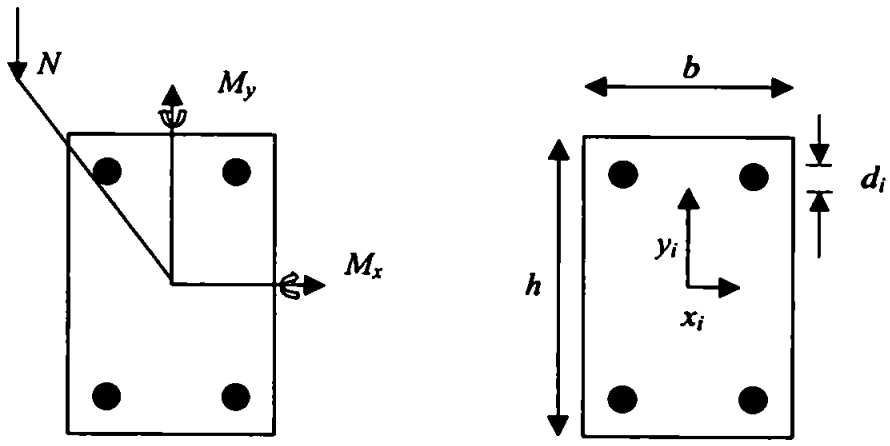


Figure 8.9: Inputs to the Biaxial Column problem: N – ultimate axial load, M_x, M_y applied design moments due to N, depth h and breadth b of the column. Decision variables: the position and diameter of each reinforcement bar i in the model: x_i y_i and d_i. See Table 8.1 for a definition of each symbol.

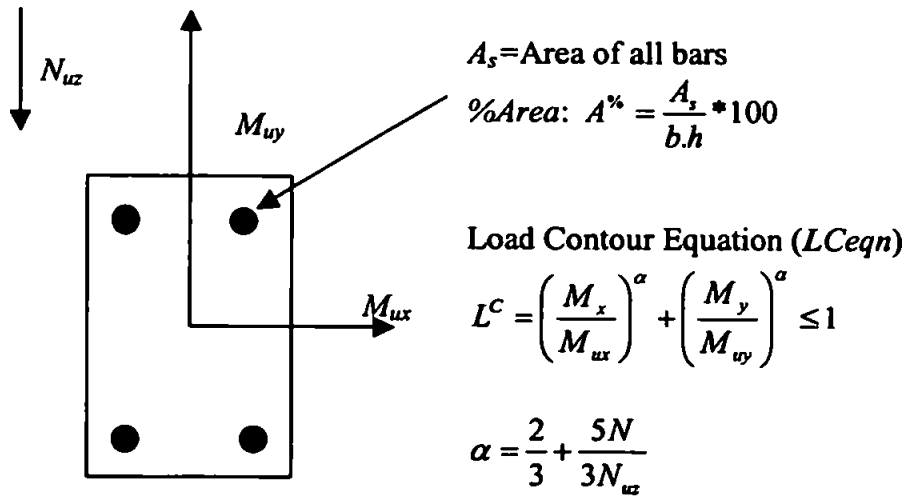


Figure 8.10: Outputs to Biaxial Column problem: M_{ux} M_{uy} uniaxial maximum moment capacity of section under action of the ultimate axial load N. The load contour constraint ($L^C = 1$ is optimal) is computed to estimate the suitability of the design according to CP110 (1972). α is found using N_{uz} (the ultimate resistance to pure axial load). The total cross-sectional area of reinforcement bars is also an output, given as a percentage of column cross-section area ($A^{\%}$).

The other important output from the model is the total cross sectional area A_s of the reinforcement bars used; for comparison and simplicity this is given as a percentage (*%Area*) of the total column area: $A^{\%}=100*A_s/b/h$). A good solution to this problem will produce a result of the load contour equation (*LCeqn*) as close to 1 as possible with A_s as low as possible. The fitness value used in the genetic algorithm could be any one of these outputs (objectives) or a combination of them.

Parameter Name	Symbol / System Name	Type
Axial load	N	Fixed input
Applied moment in x dir.	M_x	Fixed input
Applied moment in y dir.	M_y	Fixed input
Breadth of column	*b (smaller size)*	Fixed input
Depth of column	*h (larger size)*	Fixed input
Number of possible bars	*nbar*	Depends on b and h
Diameter of bar $i=1,...,nbar$	d_i / *diam*	Decision variable
Position of bar in x dir.	x_i / *xpos*	Decision variable
Position of bar in y dir.	y_i / *ypos*	Decision variable
Alpha	α	Dependent variable
Ultimate resistance to pure axial load	N_{uz}	Dependent variable
Maximum capacity in x	M_{ux} / *CapX*	Objective (max.)
Maximum capacity in y	M_{uy} / *CapY*	Objective (max.)
Result of load contour equation	L^c / *LCeqn*	Objective (max.) with constraint: $L^c \leq 1$
Total cross-sectional area of reinforced steel bars	A_s / *Area*	Objective (min.)
Percentage of used steel to column cross-sectional area	$A^{\%} = 100*A_s/b/h$ *%Area*	Objective (min.)

Table 8.1: Characteristics of parameters used in first biaxial column experiment. Note difference between dependent variables and decision variables.

Column Type	Axial load (kN)	Moment in x (kNm)	Moment in y (kNm)	Breadth (mm)	Depth (mm)
Small	950	95	65	400	300
Large	5000	2124	1000	1100	550

Table 8.2: The two sets of fixed input parameters used in the first biaxial column design problem.

8.3.2 Modifications Required to the System

In the first form of the biaxial column design experiment the breadth h, depth d, ultimate axial load N, and design moments M_x, M_y were all fixed. Two sets of fixed input parameters are considered (Table 8.2), the first set describe a small column with low axial load to be applied whilst the second set describes a larger column required to support a larger load (the parameters are taken from Rafiq & Southcombe (1998), known as columns 1 and 4 respectively).

In this problem the only decision variables were the positions and diameter of the reinforcement bars (see Table 8.1). The maximum number of reinforcement bars in each solution depends on the size of the column; the cross-sectional area of steel used should not exceed 6% of the total area of the column. The possible positions of the bars are also determined by the minimum spacing allowed between bars. So the number, diameter and possible position of bars can only take certain values that were hard-coded into the chromosome using design knowledge. The advantage of this approach is that valid configurations are generated automatically in the chromosome; the disadvantage is that the flexibility of design variables required to benefit from the interactive genetic algorithm system and clustering technique is not available. Evaluation of robustness of a solution in variable space is also difficult because there is no obvious distance metric between solutions.

The problem with trying to visualise all the variables of every design is shown in Figure 8.11. As the column is symmetrical the design can be specified as a quarter section; the other sections will be mirror images of the first. For the relatively small column (400 x 300mm) up to five bars can be placed in the quarter section due to the constraint on percentage of steel allowed, each bar has three parameters (diameter d and x, y position from the centre of the column) so there are 3*5 = 15 decision variables to be displayed.

Some of the bars have a zero diameter and so will not contribute to the objective function, but they are shown on this plot. It is difficult to pinpoint the nature of an individual as multiple designs become confused with each other. Because every third input relates to a similar parameter it is possible to duplicate them on the same plot; in Figure 8.12 the bars with zero diameter are removed and the x, y position of all the bars are shown in the main axes. The outputs (or objectives) can now be seen more clearly but this view still does not help the user determine the best configurations in terms of the fitness of the solution. Clustering in variable space returns a single solution because of the discrete variables. It is also highly likely that solutions will be duplicated because the same configuration can be described by reordering the decision variables.

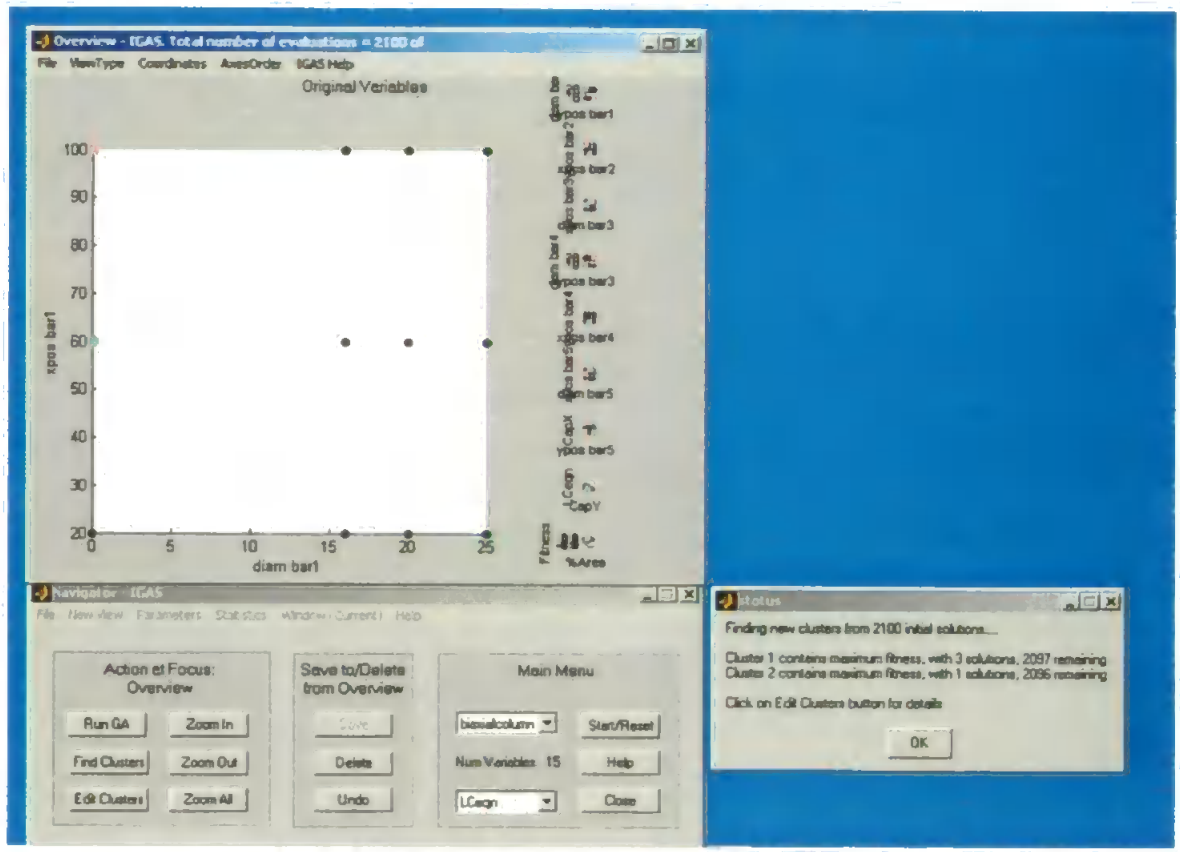


Figure 8.11: Visualisation of the first biaxial column design. There are many discrete input variables so it is not easy to visualise single designs or group of designs.

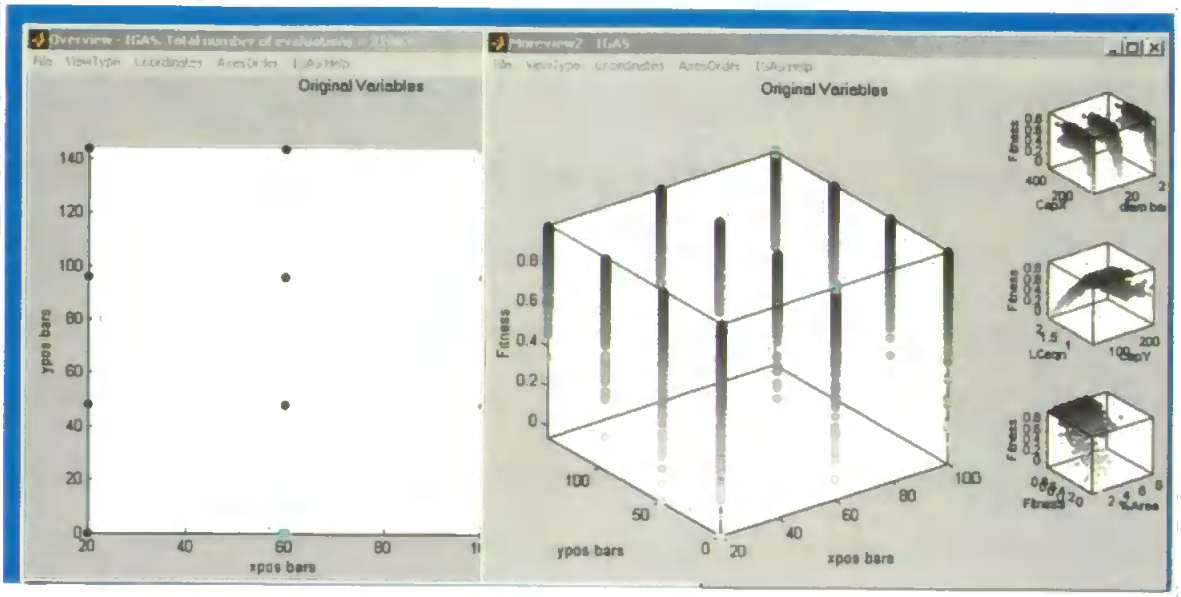


Figure 8.12: Combining similar inputs (*xpos*, *ypos* and *diam* of each bar). The understanding of solutions has not been improved.

The contributing engineer suggested it would be more instructive to visualise the solutions in objective space and have the opportunity to visualise individual designs; a number of modifications to the system were necessary to enable this. Figure 8.13 shows just the objectives in the Overview Window and another window displaying the configuration of an individual solution (available by clicking the right button near a solution). The x and y values are the positions of each bar in the quarter section from the centre of the column, the number next to the bar is the diameter of the bar d. For this column the three possible bar sizes are 16, 20 or 25mm, determined using domain knowledge (Rafiq & Southcombe 1998).

In this case the GA could use any objective or combination of objectives to generate the data. In Figures 8.11 to 8.14 the 'objective to optimise' is to force the load contour equation (*LCeqn*) as close to 1 as possible, this is achieved by maximising *LCeqn* but penalising individuals whose load contour value exceeds 1. If clustering is performed in variable space a single solution will normally be returned because the variable space is discrete; this is the solution with maximum fitness. As it is more instructive to visualise the

outputs of this objective function, features of the system were extended so that clustering and search was available in objective space and along the principal components in that space. Here the term 'clustering in objective space' means all dependent variables and objectives are used in the clustering algorithm, 'clustering in variable space' refers to just decision variables. In future it would be desirable to let the user choose exactly which variables or objectives contribute to the clustering algorithm, but for these experiments the procedure was simplified so that only decision variables or objectives with dependent variables could be used.

The rest of the clustering procedure operates as before; clusters are defined in objective parameters using kernel density estimation and the clusters containing the highest fitness are highlighted. The first two clusters found in this way are highlighted in red and green in Figures 8.13 and 8.14. The *%Area* objective takes discrete values but there are enough of them to simulate a continuous parameter so the clustering algorithm is not disrupted. Figure 8.14 shows a unimodal ridge centred around *LCeqn=1* as would be expected.

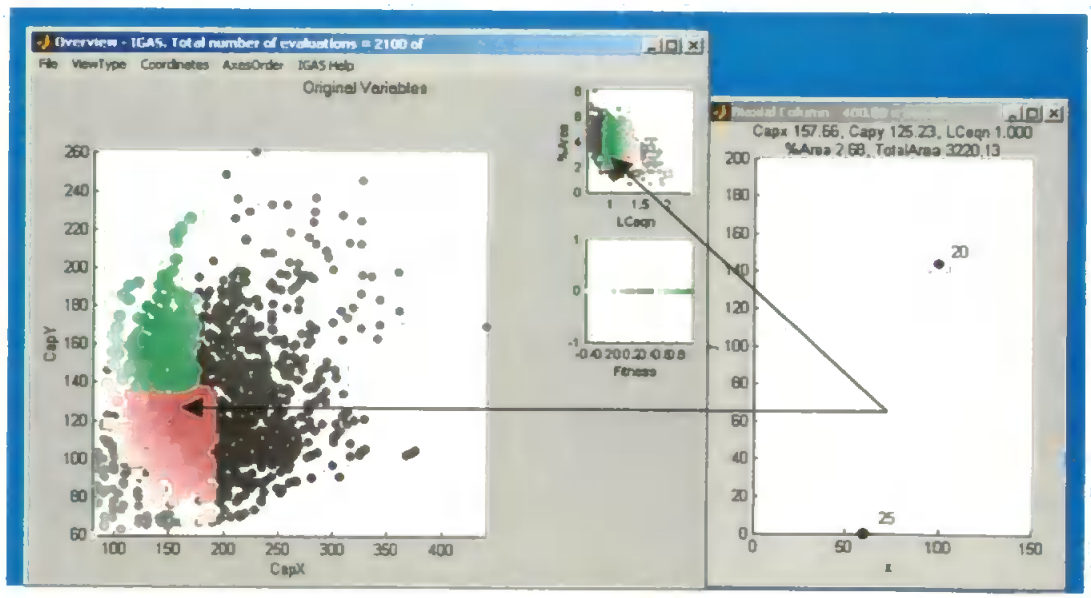


Figure 8.13: Viewing and clustering in objective space, a quarter section of an individual design is displayed by clicking right mouse button in the Overview Window. Solution of highest fitness (*LCeqn*=1) shown indicating positions (*x*, *y*) and diameter of bars. The size of the column and objective values are also given at the top.

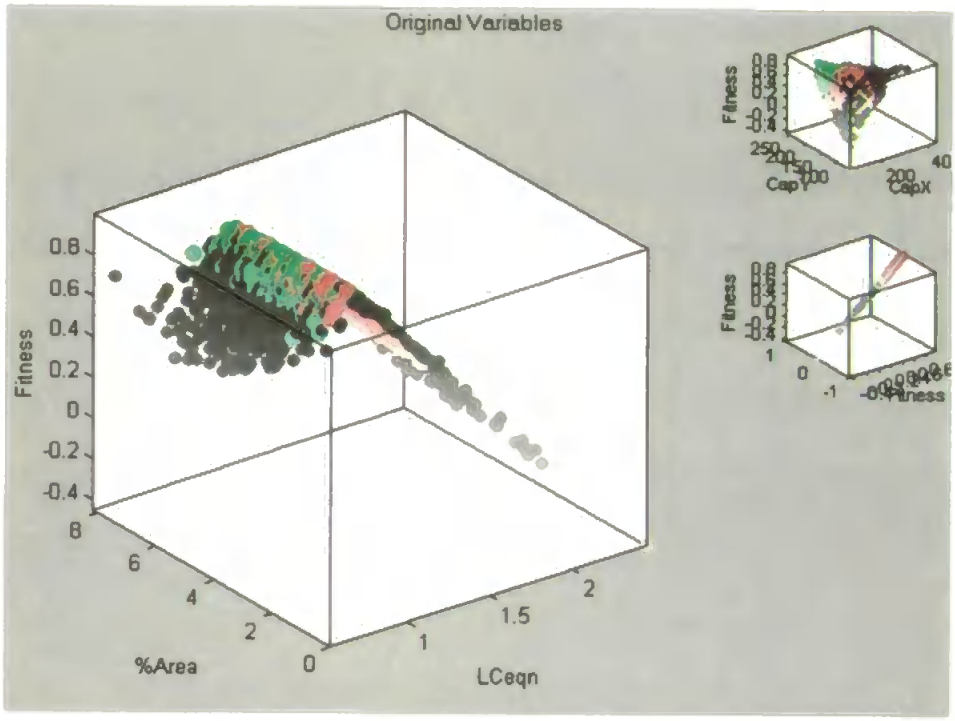


Figure 8.14: 2D+Fitness view of the objectives where fitness is maximising *LCeqn* with the constraint – a unimodal fitness landscape is revealed along the ridge of *LCeqn*=1. This time *LCeqn* versus *%Area* in main plot. Coloured clusters formed in objective space, containing solutions of maximum fitness.

Running a GA inside regions was not desirable in variable space because of the duplication of variables and redundancy in those with a diameter of zero. In this case the user is viewing the objective space and trying to find solutions that will fall inside desired regions of objective space. As stated before the main aim for the engineer in this problem is to find solutions with *LCeqn* as close to 1 as possible and *%Area* as low as possible. Figure 8.15 shows a blue cluster highlighting such solutions in the relevant two-dimensional slice of the objective space. Having highlighted this region the user can choose to run a GA 'inside' the blue region; this means that solutions falling outside this region in objective space will be penalised. Because the search space is very large it is unlikely solutions will be found inside this region by random initialisation, so solutions found inside the cluster from the previous GA run are used to initialise the next GA population.

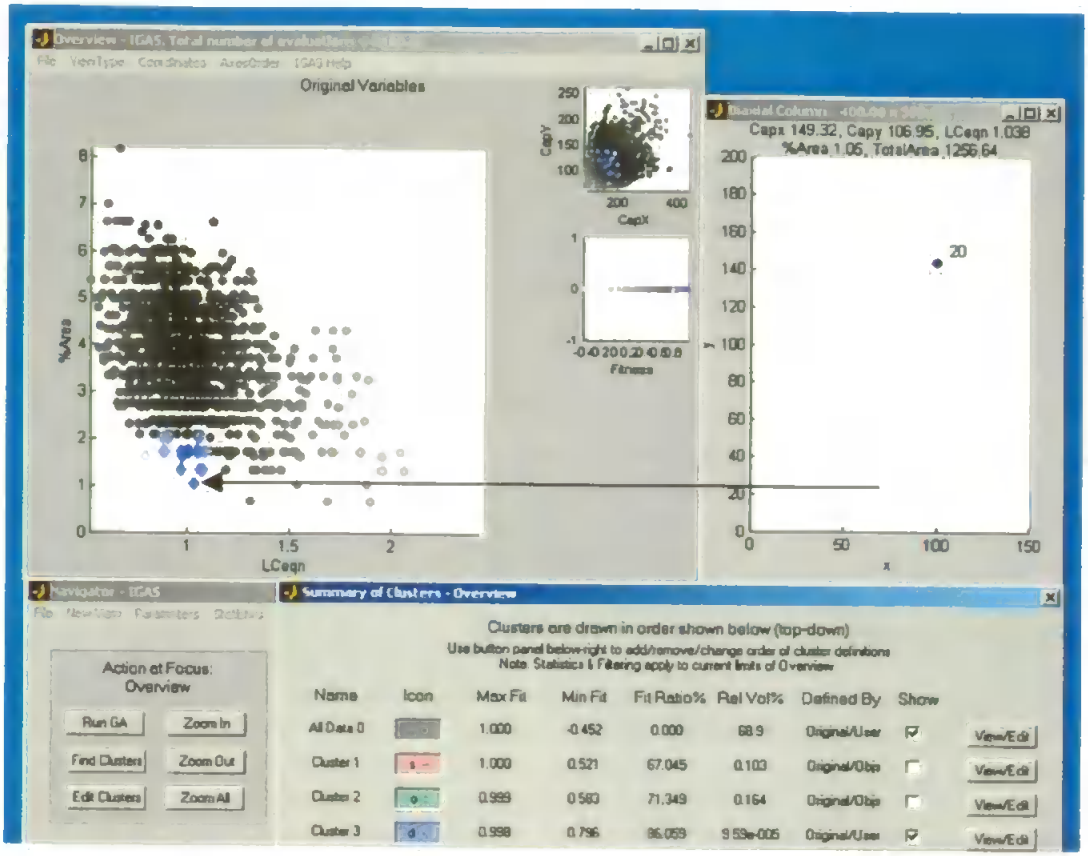


Figure 8.15: Engineer is concentrating on designs with *LCeqn* as close to 1 as possible and *%Area* minimised, the design shown is the bottom of the blue cluster - *LCeqn*=1.04 just fails constraint.

Figure 8.16 shows the result of running the GA 'inside' the blue region of Figure 8.15; many solutions have been generated outside the desired region due to the action of crossover and mutation inside the GA, however some new solutions (blue) have been discovered inside the desired region. A solution that satisfies the constraint almost perfectly (*LCeqn*=.99) has been discovered that improves on the adjacent solution found in the original GA run (*LCeqn*=.97) and reported as the optimal solution in Rafiq & Southcombe (1998). A more accurate calculation of the biaxial design moments is needed to confirm that this design meets specification; furthermore engineers may have other reasons for choosing the original design. This example shows how multiple solutions can be viewed and generated by the system, allowing the engineer to consider the pros and cons of design alternatives.

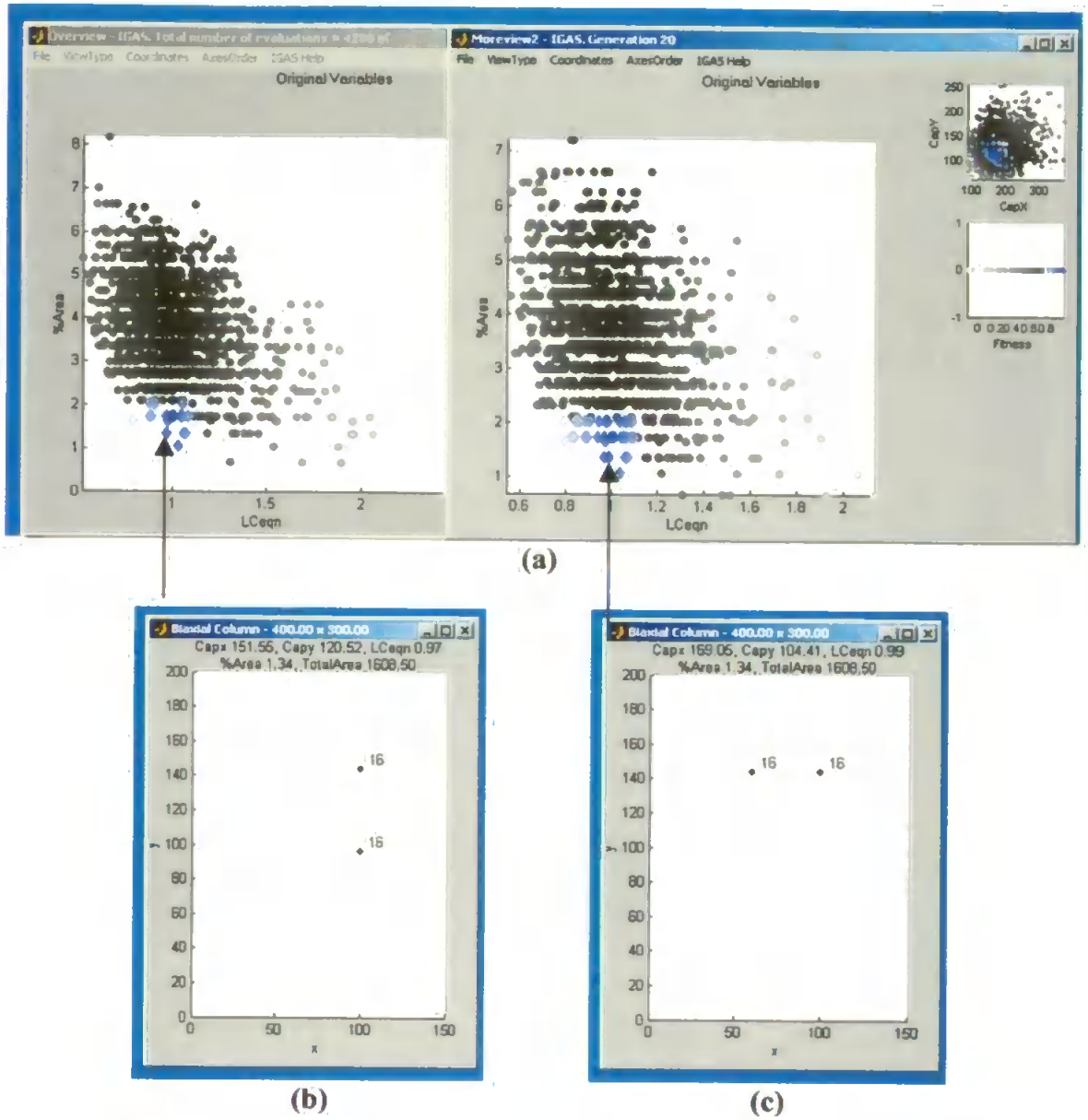


Figure 8.16: New solutions found by looking 'inside' blue region. (a) GA initialised with solutions from Overview window and any solution falling outside blue region is penalised. Solution (c) (*LCeqn*= .99) improves on solution (b) found in original GA run, although both have higher *%Area* than the solution indicated in Figure 8.15.

The use of alternative coordinate systems was also considered for this problem. Viewing and clustering the decision variables in alternative coordinates is possible, but the information is not informative because of the large number of similar variables. Moreover running the GA in an alternative coordinate system would require further modifications to the system; in the current implementation it is assumed that the alternative coordinate systems are a continuous domain, so new solutions will be generated that have no

corresponding solution in the original variable space. To solve this problem a repair mechanism or mapping function would need to be carefully designed. This work was left for a future implementation of the system.

As the objective space is more relevant to this problem and has fewer dimensions, viewing in this domain using an alternative coordinate system is potentially more informative. Figure 8.17 shows the result after combining the two lots of data shown in Figure 8.16 and clustering in the principal components. The corresponding clusters in the original variables, particularly the unusual hole in the magenta cluster, may be worth investigating. However some directly useful information is revealed by the sharp edge in the fourth principal component ('pcobj 4'). Figure 8.18 shows that this component is related to the fitness information ($LCeqn \leq 1$); if the sharp edge is coloured cyan by the user ('Cluster 6'), the corresponding cluster in the objective space is concentrated around $LCeqn=1$. The 'Summary of Clusters' dialog was modified to indicate in which domain each cluster was formed (variable, objective space or by the user).

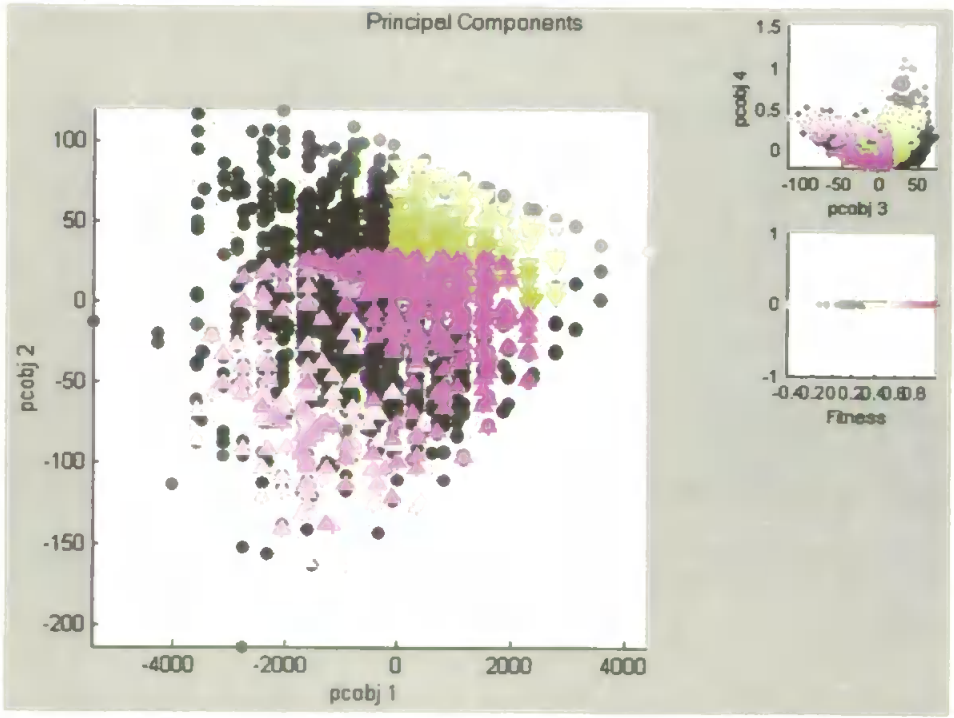


Figure 8.17: Principal component view of the combined data in objective space, natural clusters shown.

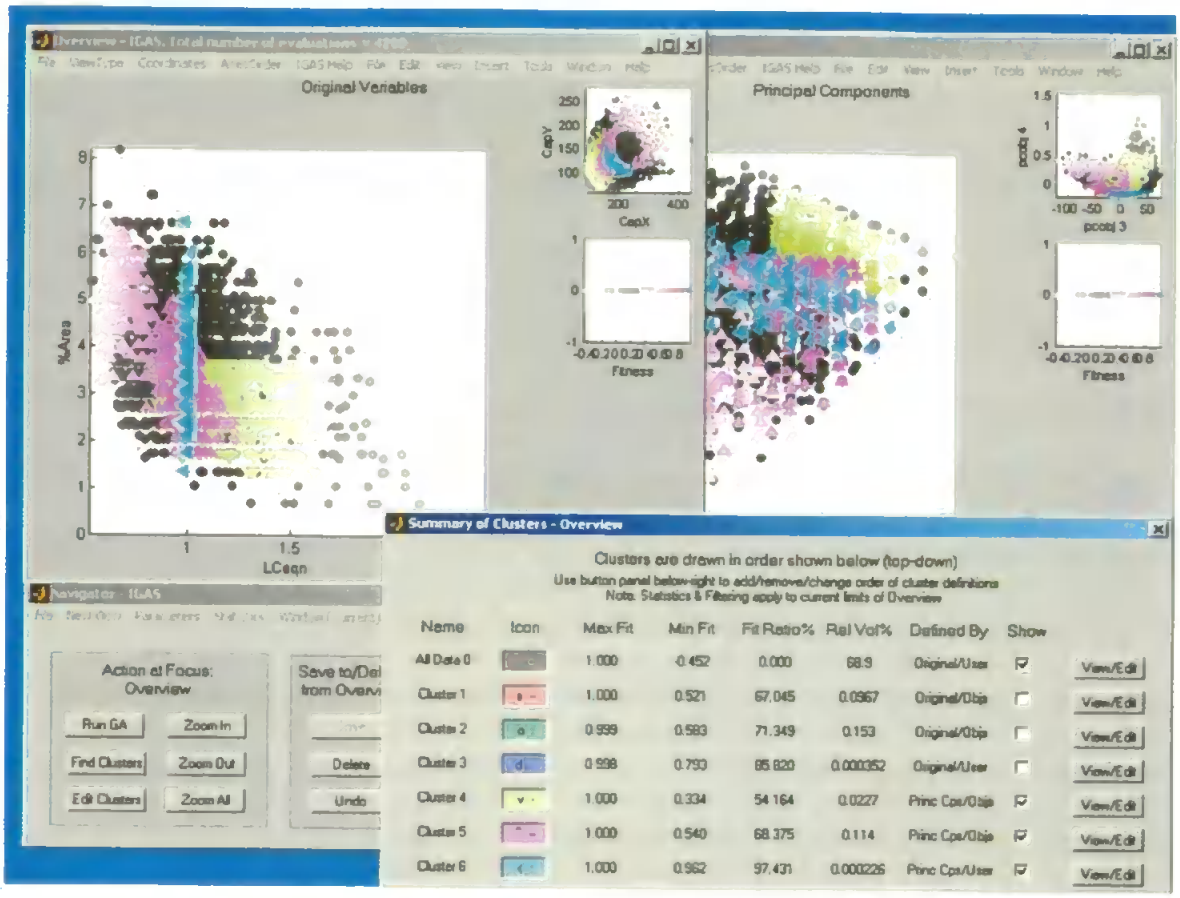


Figure 8.18: Principal component clusters viewed in objective space, cyan cluster is user-defined in fourth principal component. Summary indicates how each cluster was defined.

The sensitivity and robustness question cannot be easily solved for this problem because the inputs are discrete and the number of bars or variables contributing to the solution can vary. Therefore a distance metric between solutions is difficult to formulate, although with some thought such a metric could be designed. Even if each variable were independently tested for sensitivity, the difference in performance of neighbouring solutions would be quite large because of the discrete inputs. In theory checking the robustness of a solution is equivalent to analysing the effect of very small changes in bar position and diameter (this could possibly occur due to human influence at the detailing stage or deviations in the exact diameter of manufactured bars). The contributor and other experts did not consider such factors significant, so it can be assumed that any slight deviations are already accounted for in the model. So checking the robustness of solutions was not practical for the first biaxial column problem.

8.3.3 Large Column Design

As the size of the column grows the number of available bars and thus design choices increases, making the problem more interesting but more difficult to find the optimal design. The engineer who supplied the problem was particularly interested in comparing solutions that were on the edge of the feasible design boundary and viewing their configuration. As well as confirming the validity of the objective function to describe column design the system could also be used as a teaching tool to explain how different configurations affect the various design objectives.

Figure 8.19 shows the result after the initial 20 generational GA run for the 1100 x 550mm column. For this relatively large column, bars of larger diameter are used: 25, 32 or 40mm. Again domain knowledge is used to ensure "buildability" of the column by determining the possible discrete positions of the bars so that concrete can be placed between them. When optimising *LCeqn* a single cluster is identified in objective space. By viewing the trade-off plot between *LCeqn* and *%Area* the different possibilities can be compared. The configuration displayed is just feasible according to the *LCeqn* value but *%Area* is rather high. Such a configuration would not be used in practice, a column designer would know that better solutions exists with less bars located in the corner of the column. In Figure 8.20 a region of interest has been identified and highlighted in green, then a GA run 'inside' this region. Because the region is defined in objective space, green solutions from the Overview window are used to initialise the new GA and any solution that falls outside the green region is penalised. The result is shown in the Moreview3 window of Figure 8.20 – more solutions have been generated in the important region. A zoomed in version of the combined results (Figure 8.21) shows two solutions that have low *%Area* - the solution on the left could be improved further by moving the 25mm bar to the right.

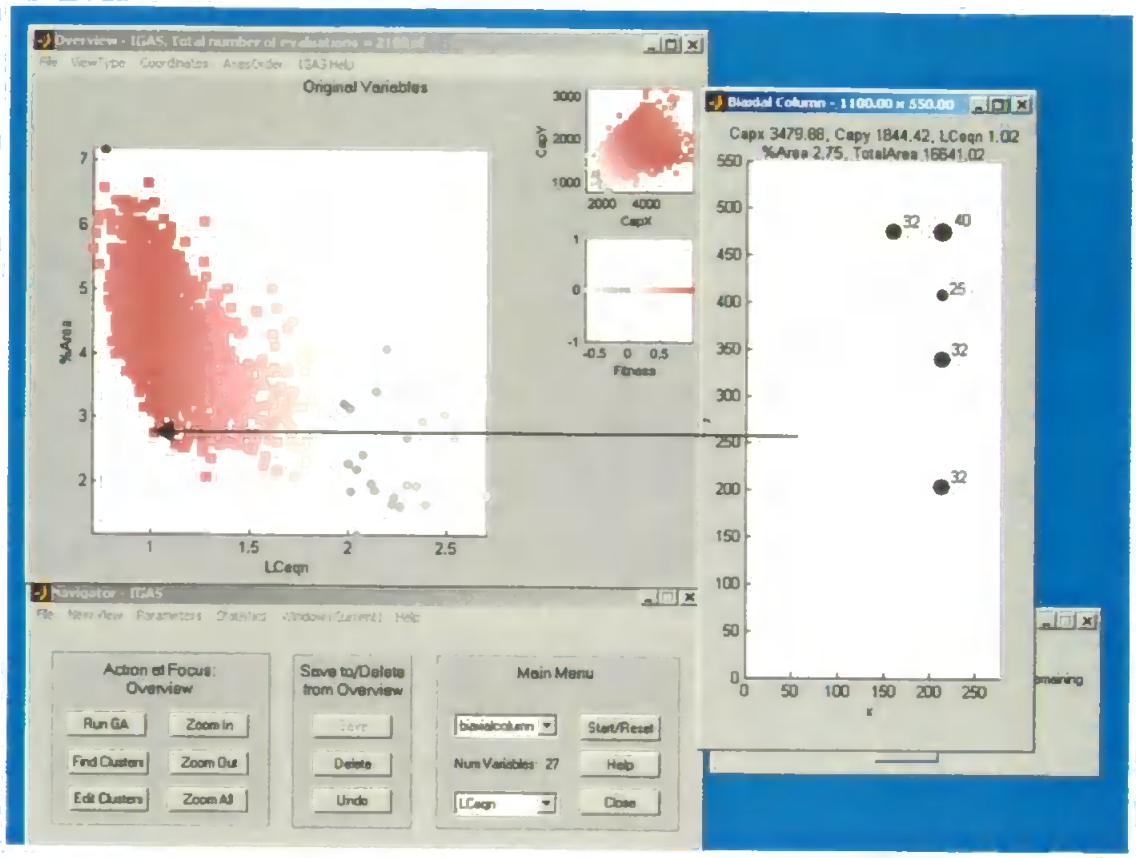


Figure 8.19: Larger column 1100 x 550mm, up to 9 reinforcement bars possible. The GA uses *LCeqn* to optimise, this solution is just acceptable at 1.02.

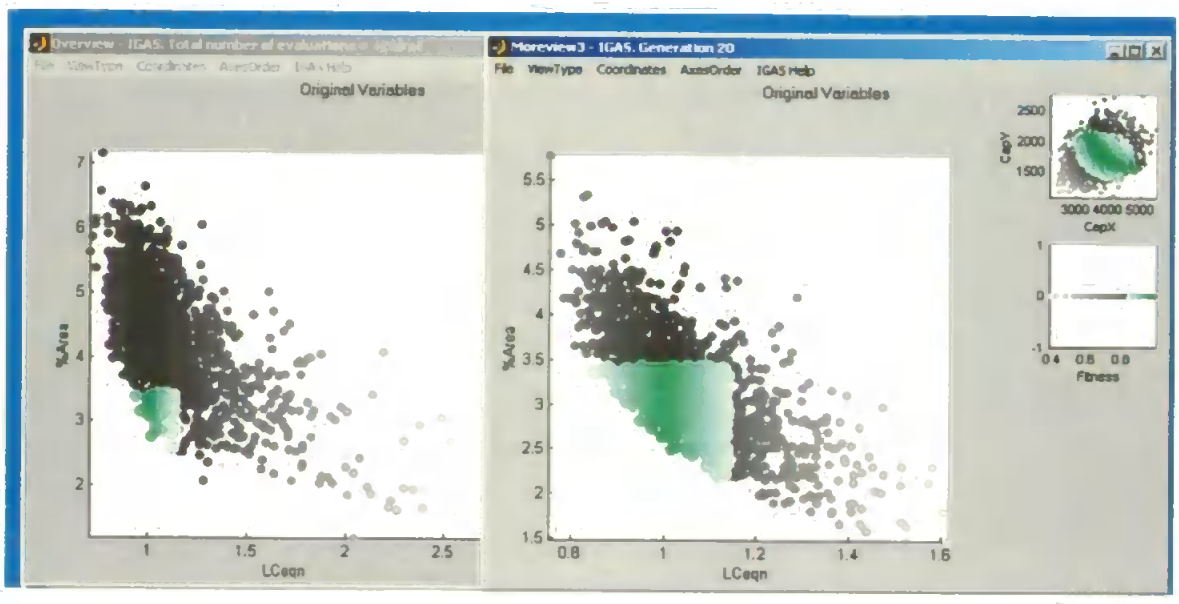


Figure 8.20: Run GA 'inside' green region, more solutions appear, increasing the choice for the user.

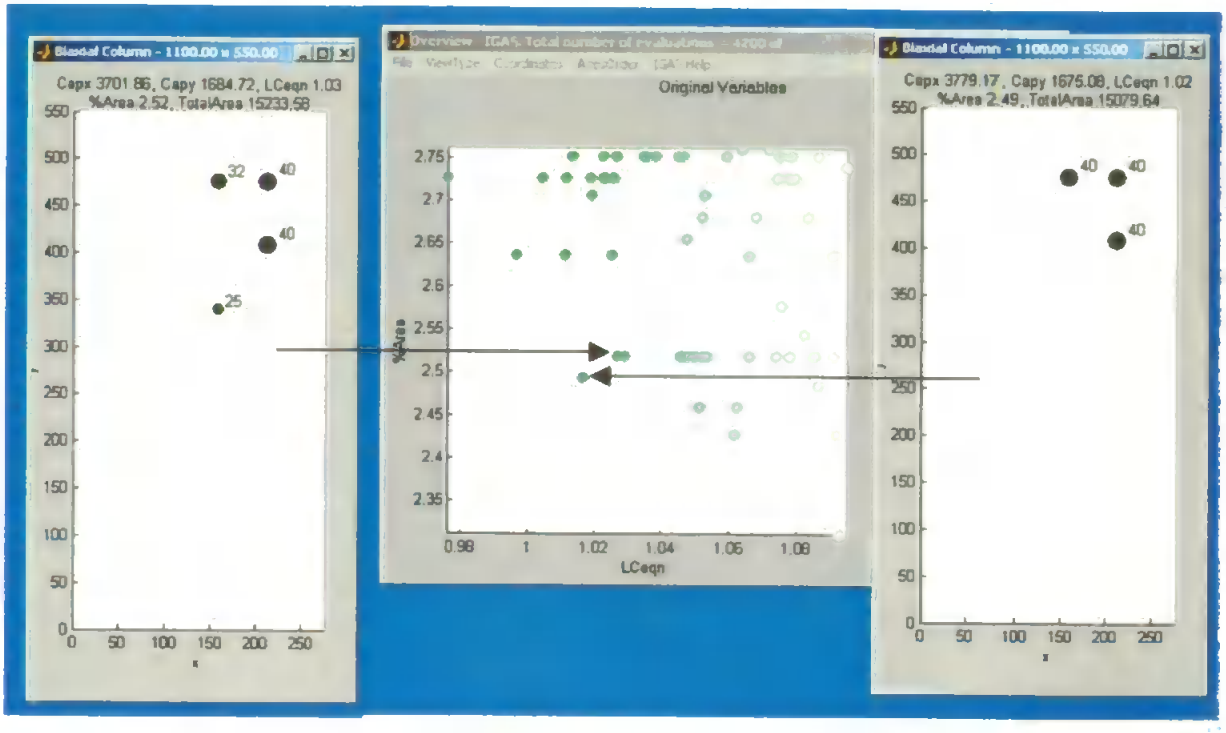


Figure 8.21: Combining the results, two good solutions found. The left solution could still be improved by moving the 25mm bar to the right.

Optimising with a different objective can give very different initial results. Figure 8.22 shows the result of minimising *%Area*, the optimal result in this case is no reinforcement bars, but this result is outside the range of the constraint. Again it is possible to highlight a region of objective space (around *LCeqn*=1) and trying to force a genetic algorithm to find solutions inside the regions. There are not many feasible solutions from the initial run, but more solutions have been generated in the desired blue region in Figure 8.23. Figure 8.24 shows the best solution after two genetic algorithm runs, this is similar to the solution found by optimising *LCeqn* (Figure 8.21); a knowledgeable engineer would rearrange the bars to achieve an optimal layout. This example shows that choice of the optimised objective has a large impact on the efficiency of the system to find solutions that satisfy the engineer's specification.

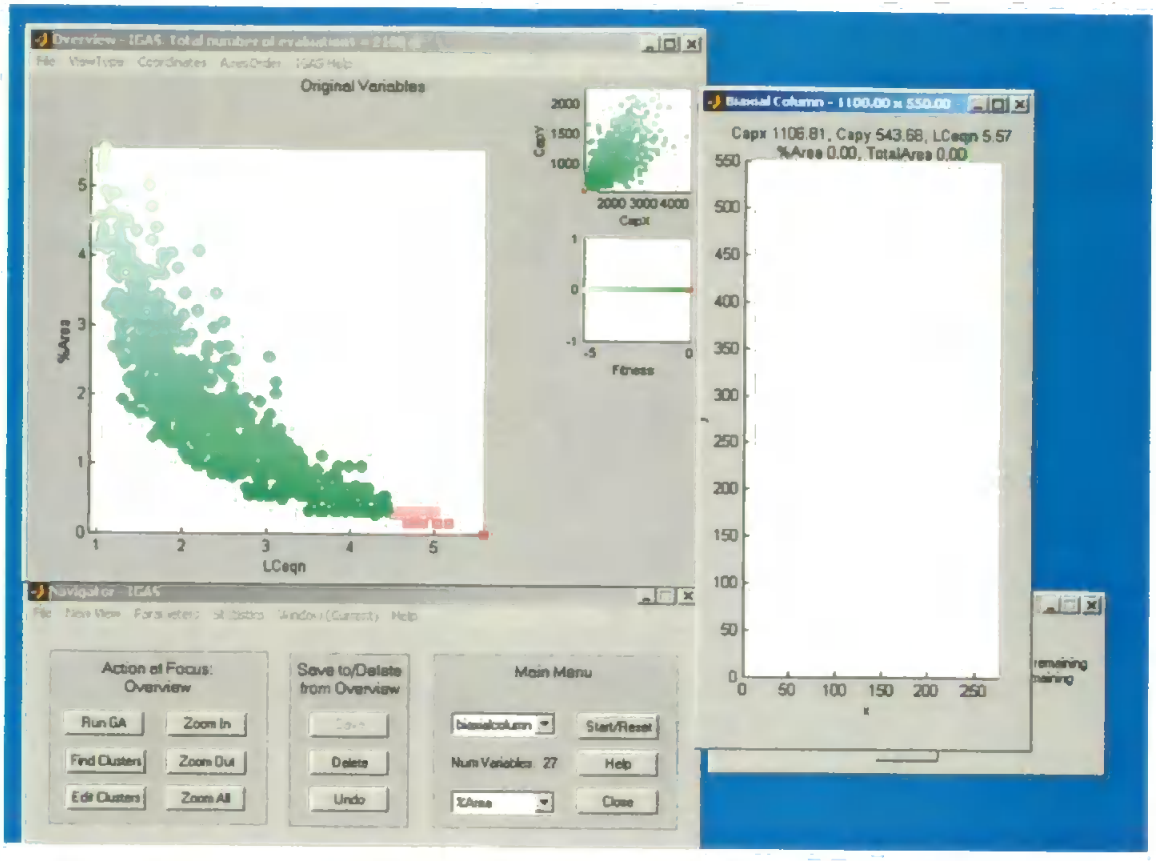


Figure 8.22: Run GA by minimising *%Area* – best solution has no reinforcement bars, but is obviously not feasible. There are not many feasible solutions available.

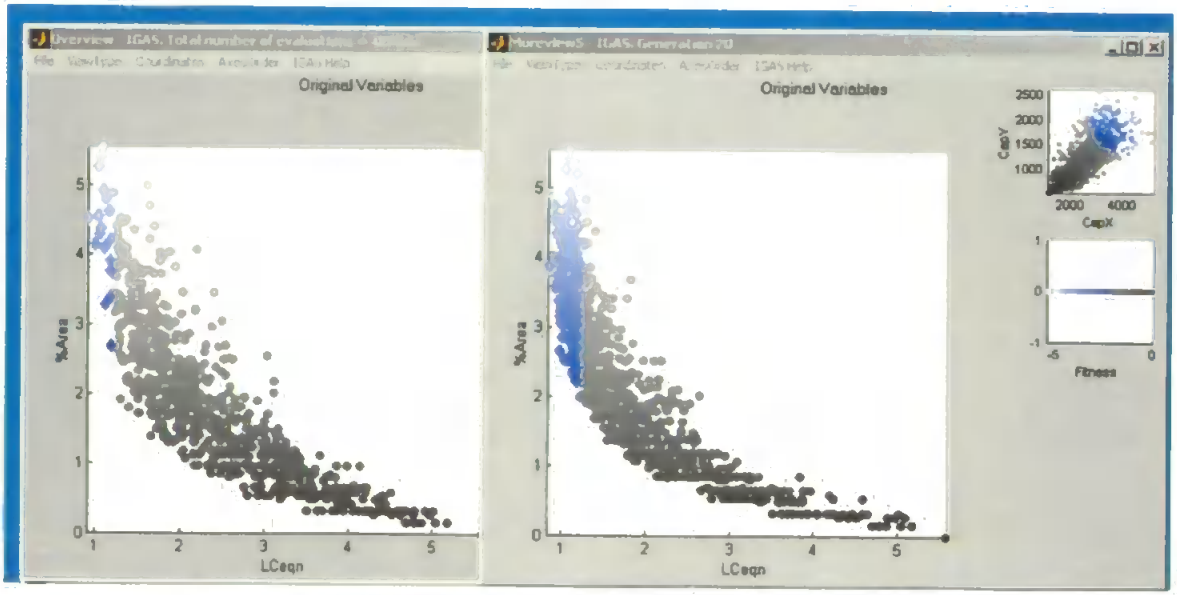


Figure 8.23: Choose a region around *LCeqn*=1, again the GA can be forced to find new solutions in the feasible region.

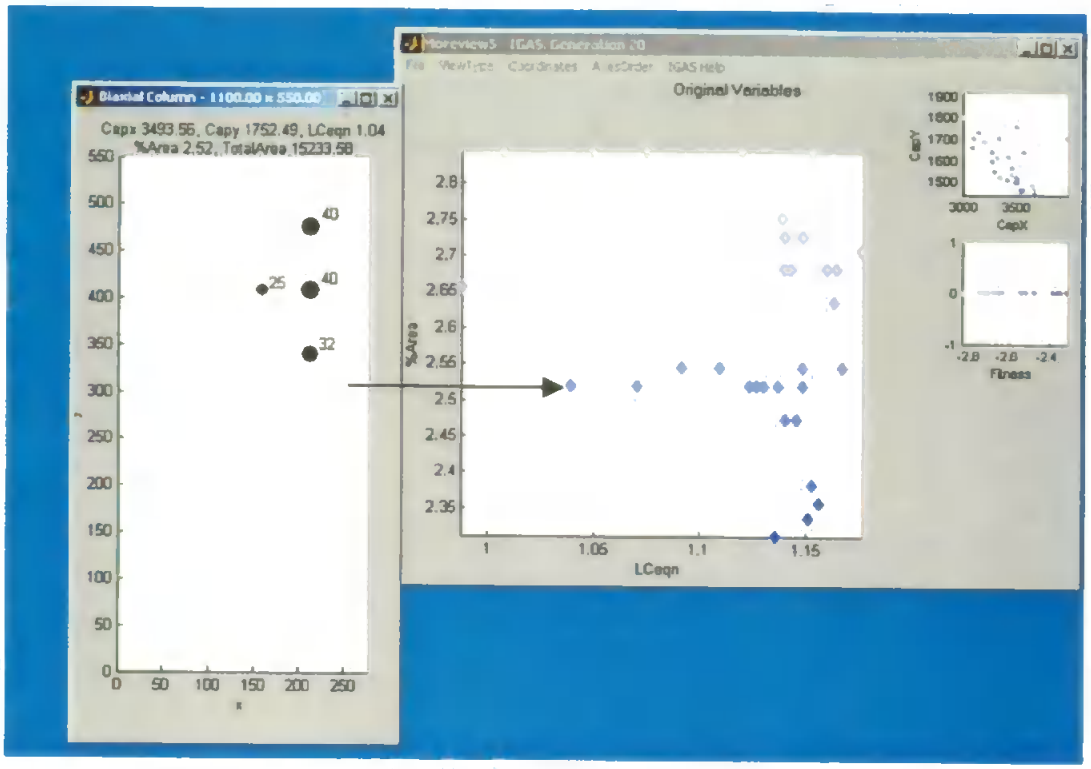


Figure 8.24: Solution shown is difficult to improve on when optimising *%Area* – converged on a similar solution to that in Figure 8.21 (left); a better solution could be found by rearranging the bars manually.

8.3.4 Multiobjective Visualisation and Optimisation

There would be an obvious benefit of viewing the data and clustering results from different objectives on the same plots, viewing multiple runs has not yet been implemented in the system, but is simulated in Figure 8.25. The trade off between the *%Area* (Figure 8.19) and *LCeqn* (Figure 8.22) objectives can be clearly seen in this figure. The relative fitness of the objectives have been normalised to aid comparison, but could be scaled to reflect the preference of the engineer. As before the user could highlight the relevant region and force the GA to search there using either objective.

However a more efficient approach is to combine the objectives in some way so that the GA converges directly on the region of interest. The two most popular techniques used in multiobjective optimisation are the 'weighted sum' approach and Pareto ranking. The weighted sum approach involves summing the objectives using weights supplied by the user. In this approach the search will concentrate on a particular part of the trade-off

curve (or response surface); this may produce undesirable results if the weights have not been chosen correctly. Another major drawback of the weighted sum approach is that it cannot resolve non-convex parts of the response surface (see Fonseca & Fleming 1995). Pareto ranking solves this problem by evolving non-dominated solutions (Goldberg 1989 p. 198, Deb *et al.* 2000) to produce the complete response surface.

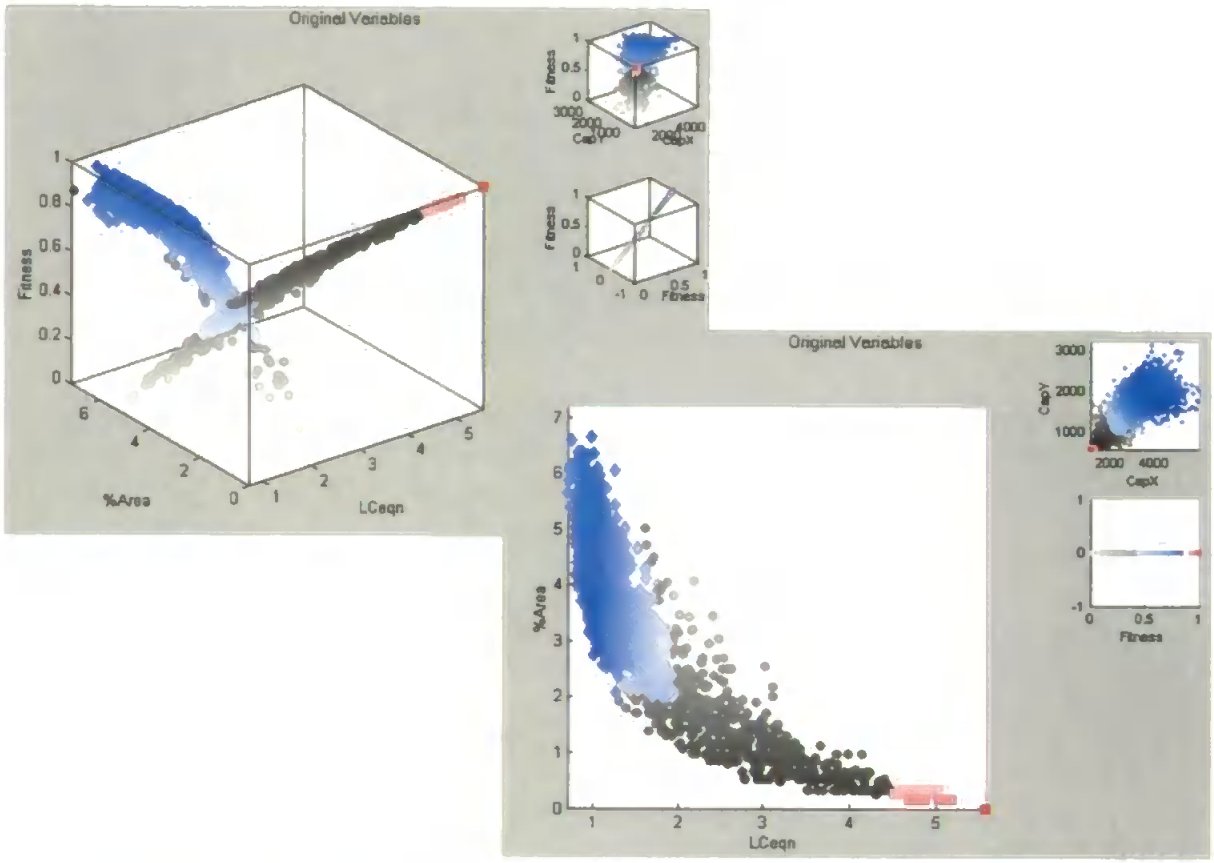


Figure 8.25: Showing two GA runs on the same plot. The data tipped by red generated by minimising *%Area*, the data tipped in blue is generated by maximising *LCeqn*≤1. Each data set has been normalised for visualisation purposes; the 'height' of each data set could be changed according to importance of each objective.

In the case of the biaxial column design problem the trade-off curve between the pertinent *LCeqn* and *%Area* objectives was known to be convex; this is confirmed by the visualisations (Figures 8.22, 8.23, 8.25 and others). Therefore the weighted sum approach was deemed to be sufficient as this problem has a well-defined region of interest. The fitness was given as:

$$Fitness = a.L^C + b.A^{\%} \qquad \textit{Equation 8.4}$$

where a and b are constants, $L^C = LCeqn$ and $A^{\%} = \%Area$ (see Table 8.1).

With a suitable choice of *a* and *b* the GA will attempt to minimise *%Area* whilst finding solutions with *LCeqn* as close to 1 as possible. Figure 8.26 shows the result of a GA using the combined fitness with weightings suggested by the contributor. This result and the result given in the right-hand side of Figure 8.21 are both superior to the results given in Rafiq & Southcombe (1998) and Rafiq (2000) in terms of steel used, although their ability to withstand the applied load needs to be confirmed.

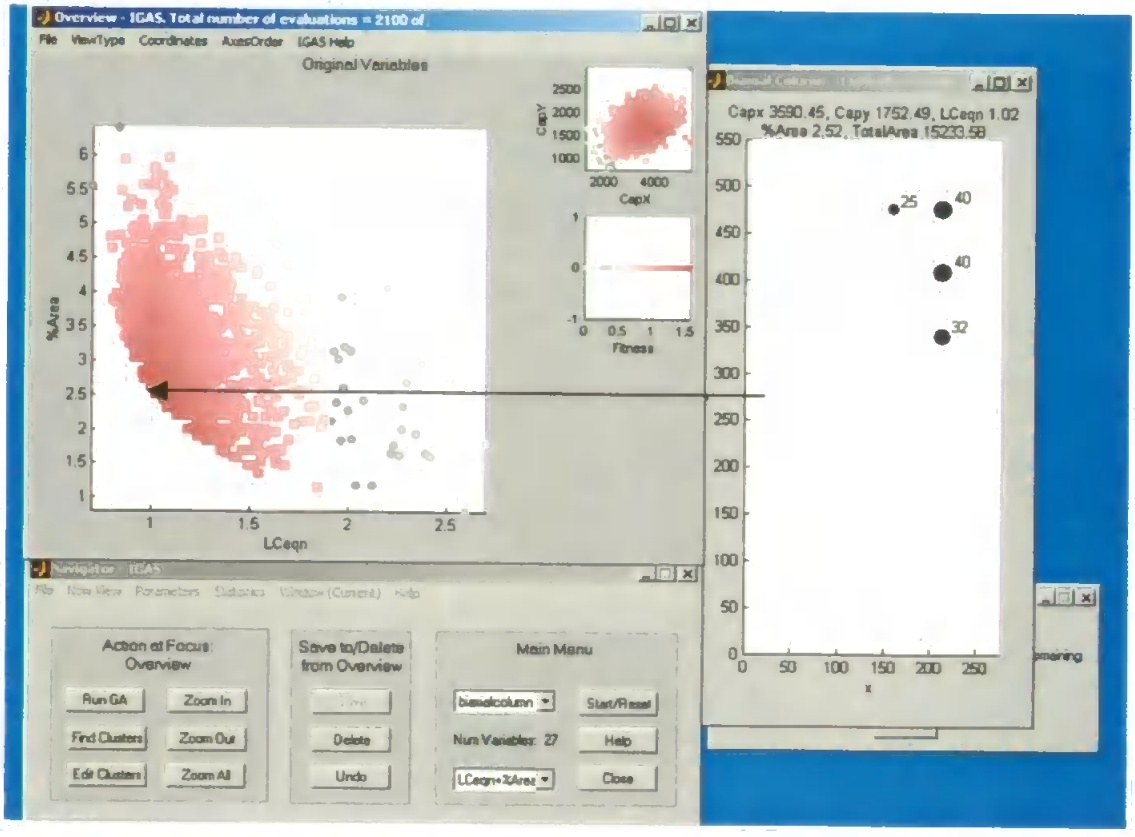


Figure 8.26: Optimising both *LCeqn* and *%Area* gives a result that improves on the interaction led results of Figure 8.21.

Figure 8.27 shows how smooth the fitness landscape is, almost guaranteeing a solution in the desired region; the green cluster contains the fittest solutions highlighted by the user in the parallel coordinate plot. Some of the solutions in the green region violate the constraint $LCeqn \leq 1$. Rafiq & Southcombe (1998) tried to avoid this problem by initially giving more weight to the constraint to ensure feasible solutions before minimising *%Area*. Alternatively infeasible solutions could be further penalised or removed altogether from the visualisation. However for this prototype system it is preferred to keep all the

information for the user to deal with at a later stage and thus avoid hard coding domain knowledge. This time highlighting the desired region (*LCeqn*≈1 – blue region in Figure 8.28) and running the GA inside returns many interesting results. The black dot on the edge of the green region (details given to the left) is the result that was missing from Figure 8.21, this may be the optimal configuration in terms of minimising steel area, although the feasibility of the solution needs to be corroborated.

The engineer may have other criteria for choosing between columns; the number of different configurations and design options reveals the true benefit of the visualisation system. The designs shown at the top of Figure 8.28 were chosen in preference to other options because all the bars are at the edge of the column, maximising the moment effect of each bar. This picture is informative, but an experienced engineer would choose designs with the largest bars in the corner of the column (Rafiq 2000). In practice bars sizes are not mixed in individual designs, so the other solution shown in Figure 8.21 (right hand side) could be optimal in this case. Further filtering of design choices could be performed manually on this plot or by manipulating the variable space.

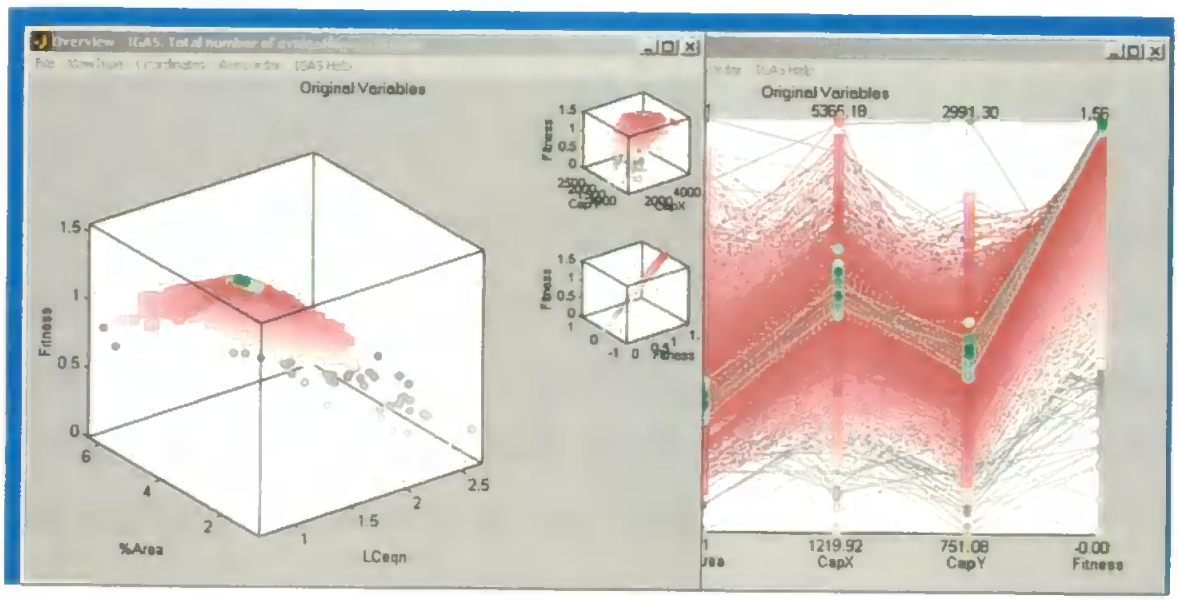


Figure 8.27: Fitness landscape view of combined objective *LCeqn* and *%Area*. Fittest solutions highlighted using the parallel coordinate view.

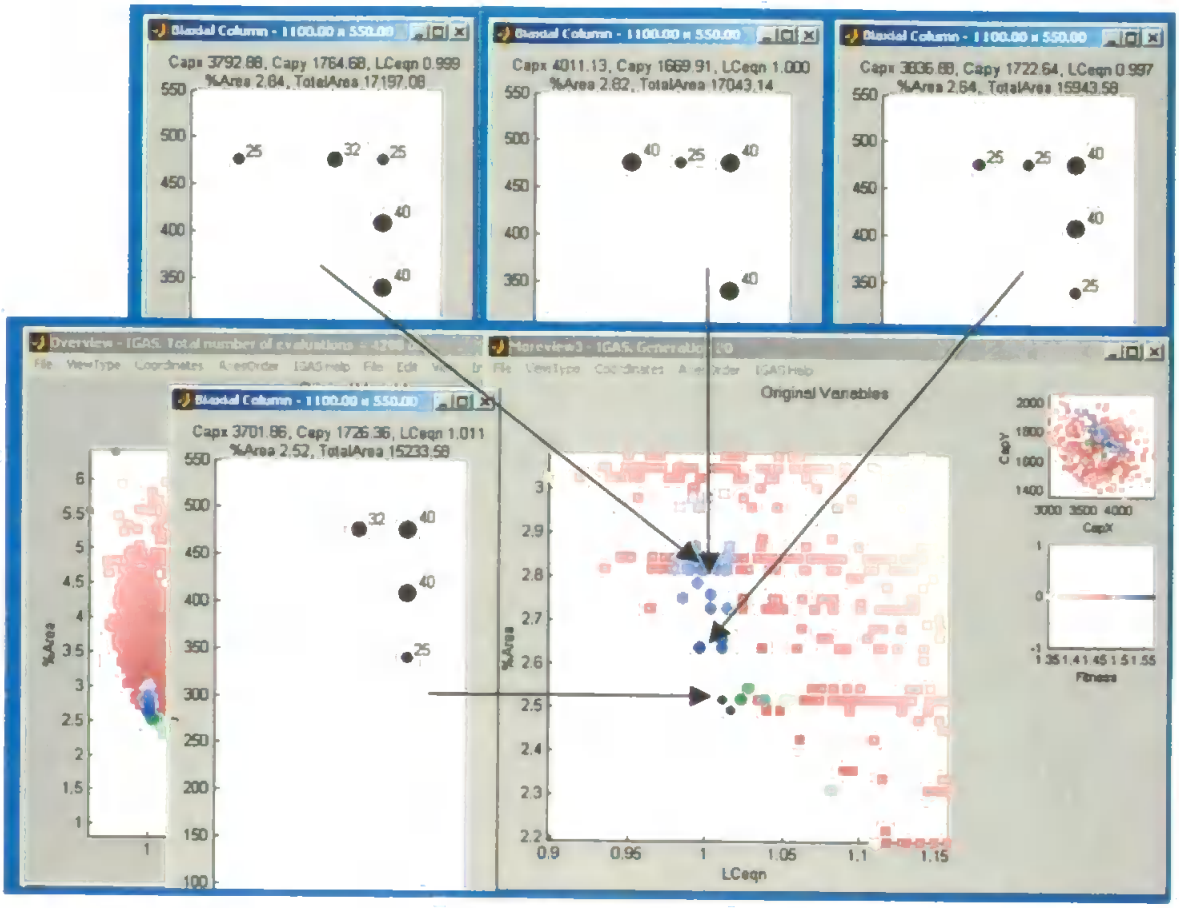


Figure 8.28: Another GA run inside the blue region generates many feasible solutions.

Another possible application of the system is as a teaching or descriptive tool. The two moment objectives M_{ux} and M_{uy} (*CapX* and *CapY* in figures) contribute directly to the load contour equation (*LCeqn*) and interesting, but intuitive, conclusions can be drawn from studying this trade off. Figure 8.29 shows the moment trade-off for the combined data found in Figure 8.28. The contributing engineer was impressed by the knowledge discovery available when interacting with this plot. Ideally both moments should be maximised but within the *LCeqn* constraint. If *CapX* is increased only, then the bars are arranged horizontally at the top of the column (Figure 8.29 right), conversely a high *CapY* is produced by arranging bars vertically (left). Low *CapX* and *CapY* result from bars being placed too close to the centre of the column (Figure 8.29 bottom), whilst an *LCeqn* value of 1 is achieved by placing the bars at the corners of the column. However finding the blue optimal solutions that minimise cross-sectional area would be a case of trial and error using this visual-interactive method alone.

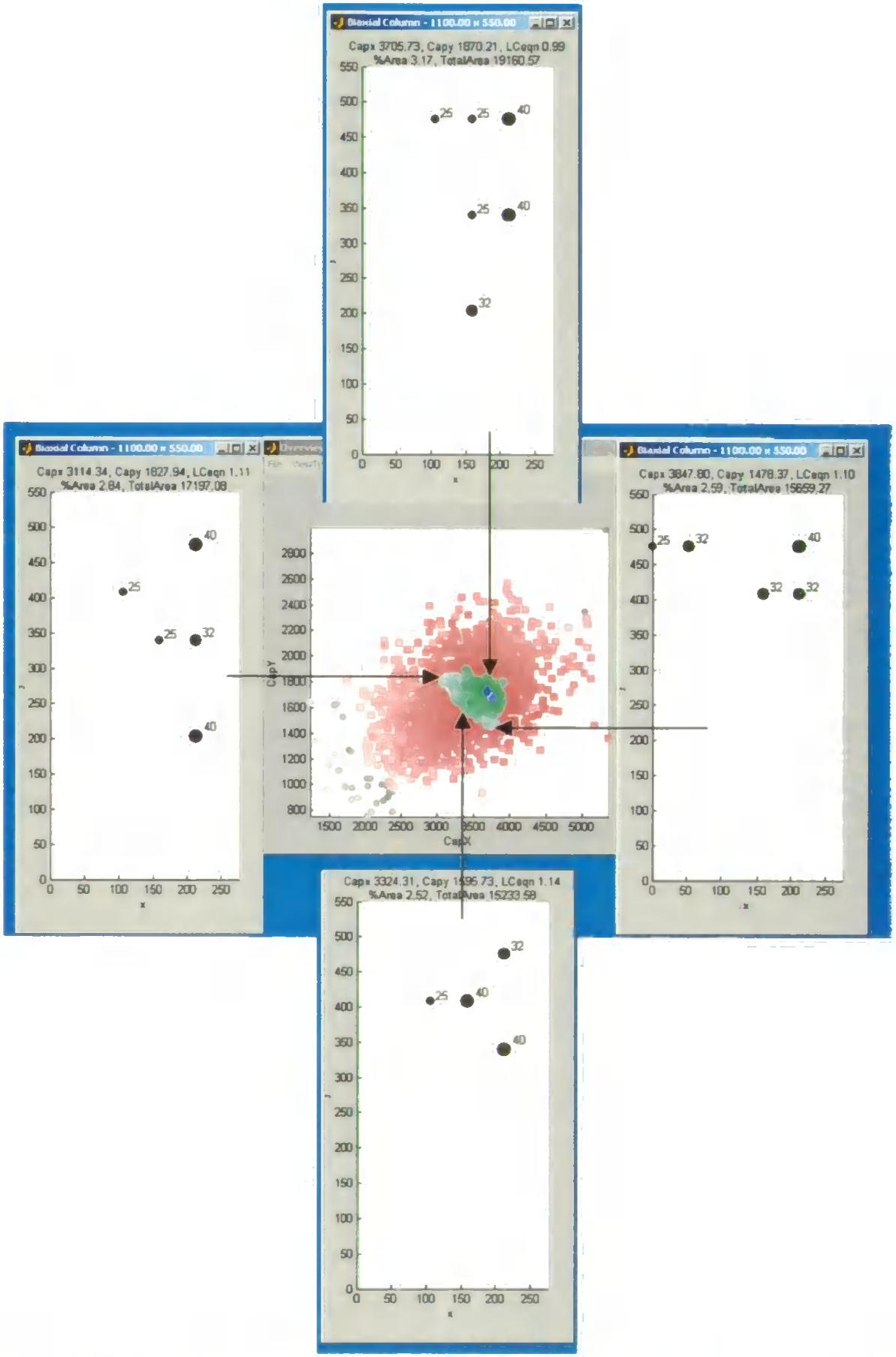


Figure 8.29: Looking in *CapX* against *CapY* is also instructive – configurations change as the user moves around the trade-off picture. This will confirm the robustness of the system and objective function to an experienced engineer and is valuable knowledge discovery for a novice user.

8.3.5 Conclusions

The first biaxial column design problem presented a number of challenges to the system, underlining the fact that real world problems are usually not well-behaved, predictable functions. The main problem was that the inputs to the model were the size and position of each reinforcement bar, some of which could take on zero diameter. Visualisation of individual solutions and groups of solutions were difficult and the clustering algorithm tends to return just a single solution because of the discrete nature of the inputs. The system was modified by overlaying all the bar inputs onto three variables and ignoring the bars with zero diameter, but differentiating between individuals and clusters of solutions was still a problem. Instead the solutions were viewed in objective space and the clustering and GA behaviour modified to work in this space using a penalty function. Such modification is not an ideal approach; a better method would be to find a smooth mapping so that feasible solutions would be more easily generated.

For this function the automatic clustering in objective space highlights the fact that each objective is unimodal. The principal component view again showed interesting behaviour in the data that requires further investigation. However allowing the user to define a region in objective space and then force the GA to search inside this region was immediately successful. Best results were found by initialising the GA with solutions already found inside a region and penalising solutions that fall outside the region. Looking at the configurations of neighbouring solutions in objective space encourages knowledge discovery and understanding of the problem.

Optimising different objectives with the GA concentrates solutions at either end of the trade-off spectrum. Another natural extension to the system was simulated by allowing comparison of results from different objectives on the same plot, the user could define a region that all objectives should try to satisfy. Interaction can help to force the GA to

search in the desired region, but alternative techniques such as weighted sum or Pareto ranking are more efficient. A combination of the weighted sum approach and user interaction indicated the large number of design alternatives generated by the system. Nevertheless the use of the Pareto front is at the forefront of multiobjective optimisation research (Fonseca & Fleming 1995, Deb 2000, Rafiq 2000) and would be the subject of future research for the system. In particular optimising and viewing the fitness landscape in terms of the 'Pareto rank' would be interesting as more designs should be generated along the desired trade-off between objectives and relevant feasible designs would be more efficiently found.

There is no advantage in using the negative GA in objective space as the positions of degraded solutions can already be seen and such a search will not provide any information about the sensitivity and robustness of solutions. Evaluating sensitivity in variable space is problematic because it is necessary to know the neighbourhood of solutions in variable space. Each design choice is distinct and because of the representation it is difficult to relate designs. This situation could be improved by pre-processing the order of variables for each individual so that solutions can more easily grouped in terms of numbers of bars or maximum diameter of bars. In practice an engineer would look at a design generated in the feasible region of the objective space and evaluate its robustness using domain knowledge. Given a configuration of mixed 40mm and 25mm bars, for example, the engineer would have a feel for the effect of moving or swapping bars and know that the larger bars should end up in the corner to improve the efficiency of the design. As many infeasible or impractical designs are generated by this function it may be worth introducing further domain knowledge into the problem.

As mentioned at the end of Section 8.3.2 the robustness issue is not significant for individual designs as sensitivity due to the exact positions and size of bars is already

accounted for. This was confirmed when the contributing engineer received feedback from experts in the column design field; they stated that in practice a column is over-designed to ensure it complies with building and safety issues. These issues were seen as so fundamental to column design that the contributing engineer designed an alternative objective function. As the problem seemed more suitable for the interactive system, further research was carried out on the modified function as described in the following section.

8.4 Biaxial Column Design 2

8.4.1 The Need for Another Biaxial Column Design

In theory the columns found in the first biaxial column experiment (Section 8.3) are the most efficient and will withstand the applied load with minimal amount of material used. But in practice column design experts will ensure safety by simplifying the manufacturing process (called bar detailing), using one bar size and reducing the possible choices for placing the bars. The default design is to place reinforcement bars around the edge of the column, including the sides, to stop it 'exploding' due to a large axial load. These safety factors on column design are highly recommended by practitioners and common practice in the UK and elsewhere, so the contributor coded the constraints directly into the objective function.

Thus domain knowledge was increased and design options decreased in the new problem. However the designer still has a number of decisions to make due to the "buildability" of the column. The columns are built using a form and the bars are held in place by shear links. The arrangement of the bars has an impact on the ease of detailing the column and keeping the bars in place with the links. Optimal arrangements are difficult to describe and depend on size and number of bars present in the column, so the more design choices that can be generated the more likely an experienced engineer would be able to choose a safe and easy to build column.

The axial load N and design moments M_x and M_y are fixed in this model, but the column depth (h) and breadth (b) are allowed to take certain discrete values. The other variable inputs are the number of reinforcement bars (whose positions are determined by the depth and breadth of the column) and the diameter of all bars (12, 16, 20, 25, 32 or 40mm). As well as the objectives listed for the first biaxial column design, the cost of making the column is also an objective. The cost is a sum of the cost of concrete, steel and form used to build the column. The required links between reinforcement bars are also returned as dependent variables (see Table 8.3 for details) and are used in the calculation of cost. Fixed input parameters for the column design problem discussed in this section are given in Table 8.4.

Parameter Name	Symbol/System Name	Type
Axial load	N	Fixed input
Applied moment in x dir.	M_x	Fixed input
Applied moment in y dir.	M_y	Fixed input
Concrete cost per m. length	*ConcCost*	Fixed input
Steel cost per m. length	*SteelCost*	Fixed input
Formwork cost per m. length	*FormCost*	Fixed input
Breadth of column	*b / breadth-col*	Decision variable
Depth of column	*h /depth-col*	Decision variable
Number of bars	*nbar / num-bar*	Decision variable
Diameter of bar	*d / bar-diam*	Decision variable
Space between bars x dir.	*space-x*	Dependent variable
Space between bars y dir.	*space-y*	Dependent variable
Diameter of links	*link-diam*	Dependent variable
Space between links	*link-space*	Dependent variable
Alpha	α	Dependent variable
Ultimate resistance to pure axial load	N_{uz}	Dependent variable
Maximum capacity in x	M_{ux} / *CapX*	Objective (max.)
Maximum capacity in y	M_{uy} / *CapY*	Objective (max.)
Result of load contour equation	L^c / *LCeqn*	Objective ≤ 1
Total cross sectional area of reinforcement bars	$A^{\%} = 100 * A_s / b / h$ *%Area*	Objective (min.)
Cost of column per m. length	*CostofCol*	Objective (min.)

Table 8.3: Parameters used in second biaxial column experiment.

Column Type	Axial load (kN)	Moment in x (kNm)	Moment in y (kNm)	Cost of Concrete	Cost of Steel	Cost of Formwork
Large	5000	2100	1000	60	750	20

Table 8.4: Fixed input parameters used in the second biaxial column design problem.

There are fewer, more general, decision variables for the new function that should respond more readily to the clustering procedure, in particular column depth and breadth. In theory it should be possible to form and view clusters described in both the variable and objective space or transform to a new coordinate system in both spaces and define clusters from there.

8.4.2 Objective Robustness

Although robustness of an individual column is catered for by over-designing of practical designs, it was hypothesised that some theoretical comparison about the robustness of objectives due to changes in decision variables could be made in this new function because clustering in variable space is more realistic than the first biaxial column design function. Figure 8.30 shows the design and objective space whilst optimising *LCeqn* (maximising but ≤ 1) in the second biaxial column function. Two clusters have been found in variable space partitioned most obviously by the *bar-diam* variable (top right hand plot). Attempts to evaluate robust regions reveal that the green region is a lot more robust than the red (Figure 8.31). The red region is made up of solutions with large bars (32mm diameter), whilst the green designs contain small bars (16 or 25mm) and is more robust in terms of *LCeqn* (Figure 8.31). This implies that large columns with small bars may be more robust than small columns with large bars. In theory if the diameter of the bar is large, changing the number of bars will affect the solution a lot, although this is not likely to happen in practice.

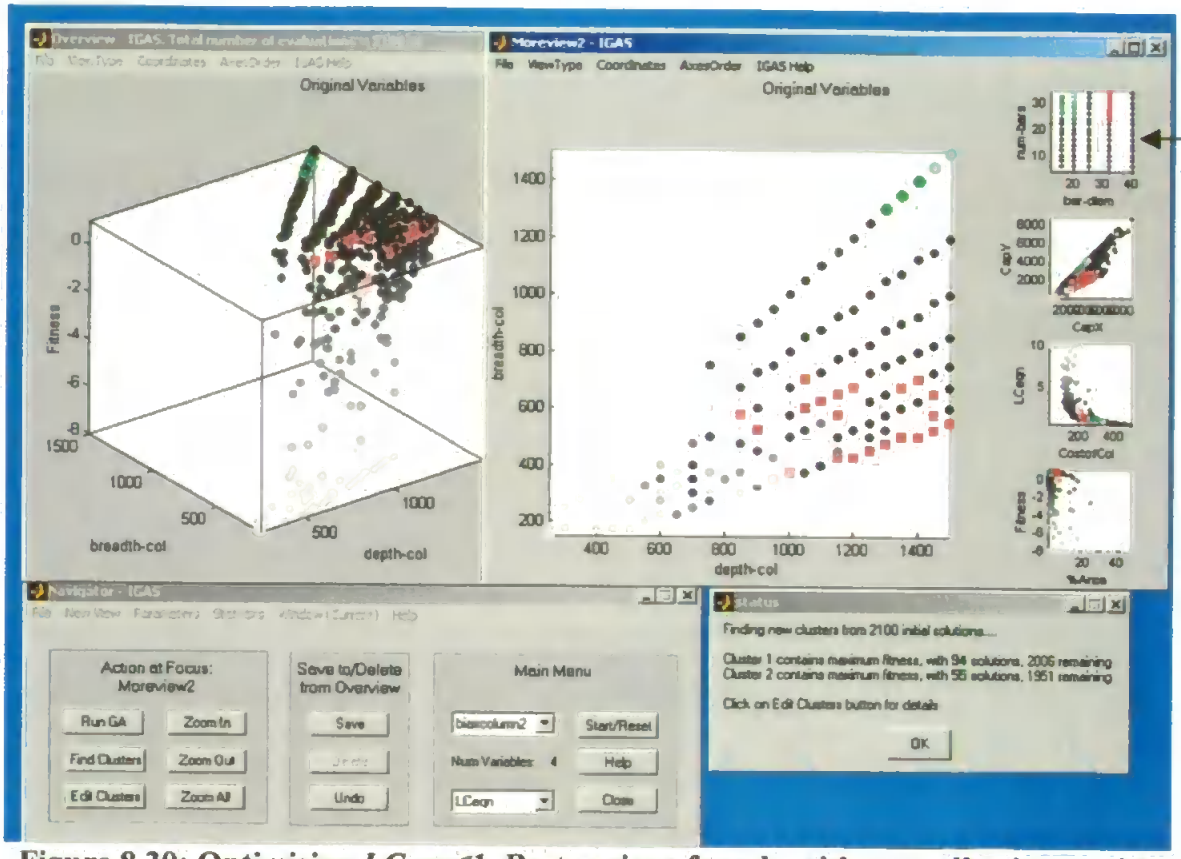


Figure 8.30: Optimising *LCeqn*≤1. Best regions found – either small column with large reinforcement bars (red), or large column with small reinforcement bars (green).

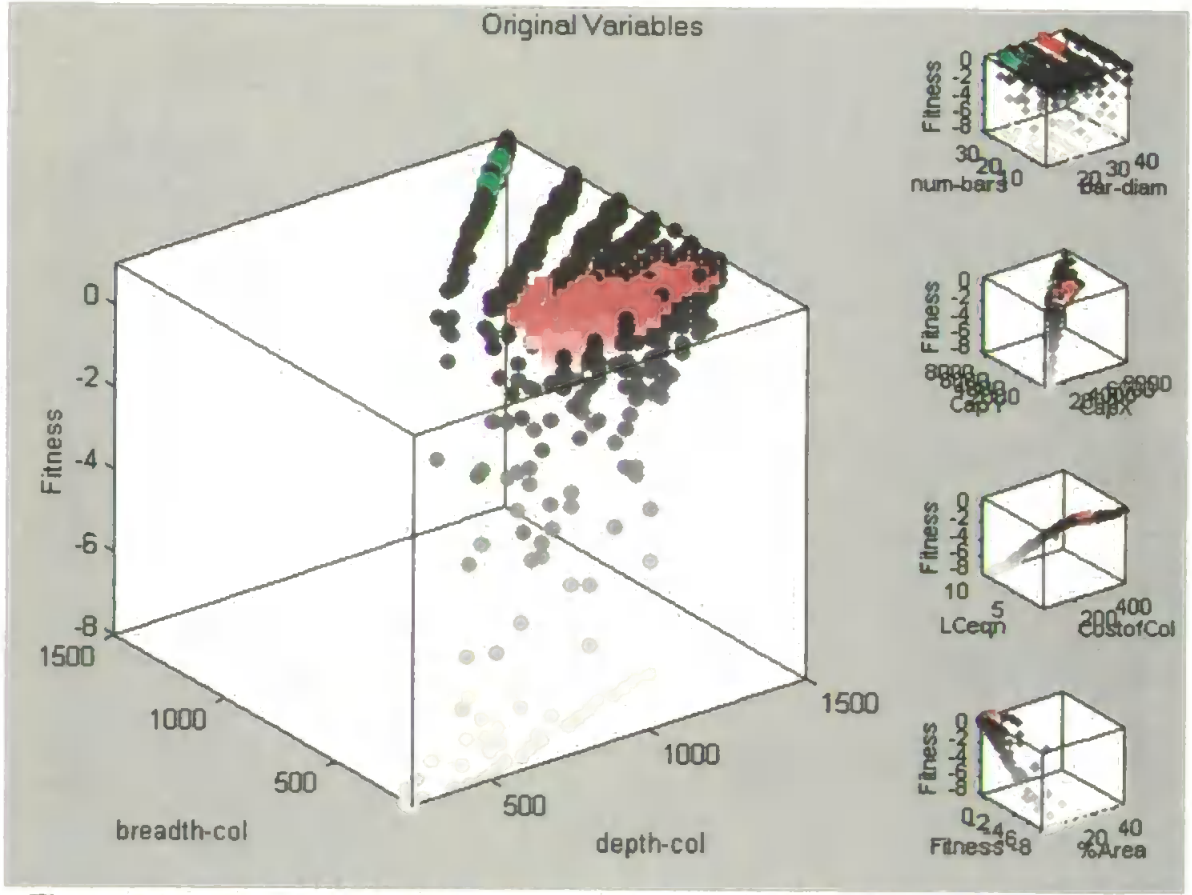


Figure 8.31: Choosing top 10% and running negative GA, large column with small bars seem to be more robust, although an individual solution in the red region may be acceptable.

Clustering in objective space reveals a number of solutions with *LCeqn* value between 0.5 and 1 (blue and yellow clusters in Figure 8.32) but the solutions cover a large portion of the variable space. A negative GA in these regions would show that there are many bad solutions because the design space is very flat although noisy (Figure 8.32). Conversely there are a number of diverse design options that produce good solutions. Clustering can again be performed in the principal component version of the variable or objectives (Figure 8.33). Analysis of these results may reveal some interesting features of the data, but a major hindrance to this analysis is the difficulty in explaining the results to engineers without knowledge of principal component analysis.

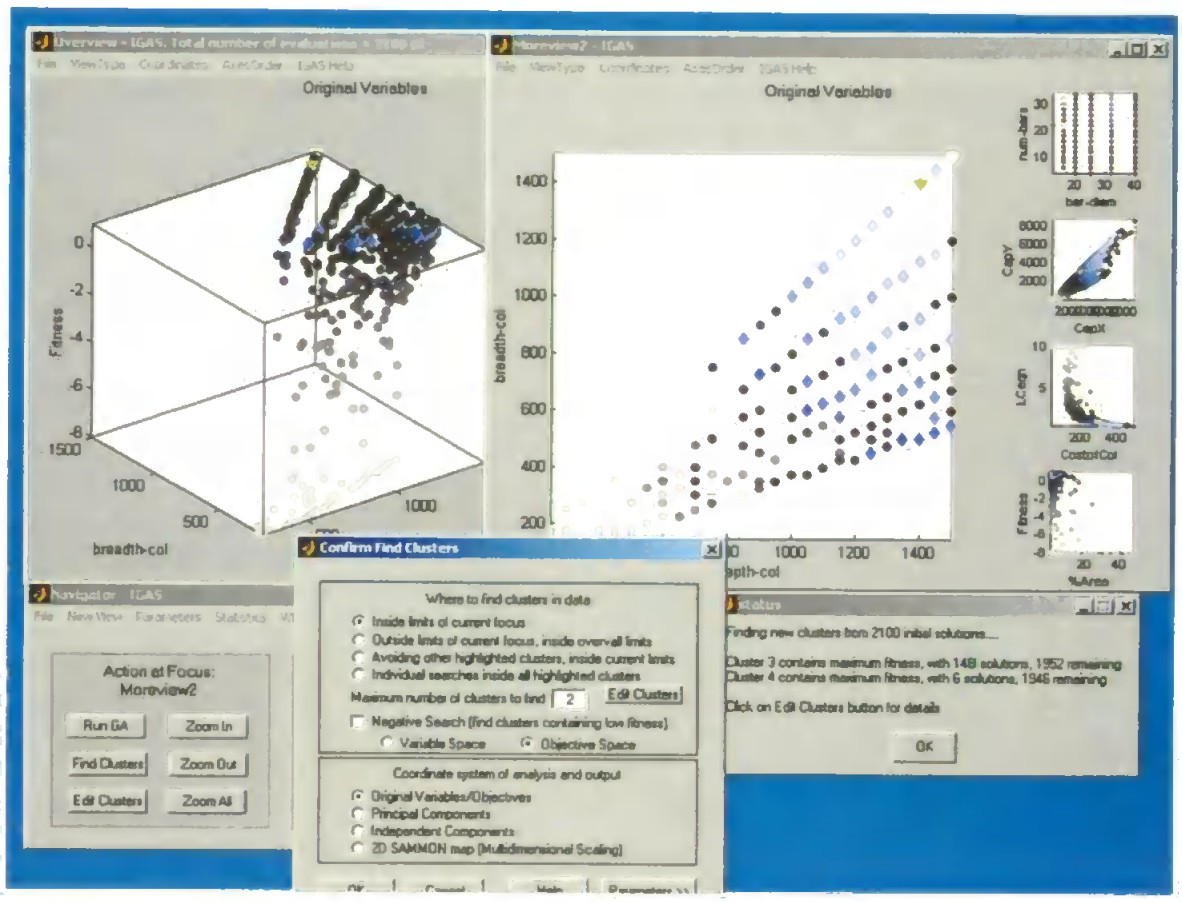


Figure 8.32: Clustering in objective space, blue and yellow solutions all have *LCeqn>0.5*. A lot of good designs found with a variety of different variable values.

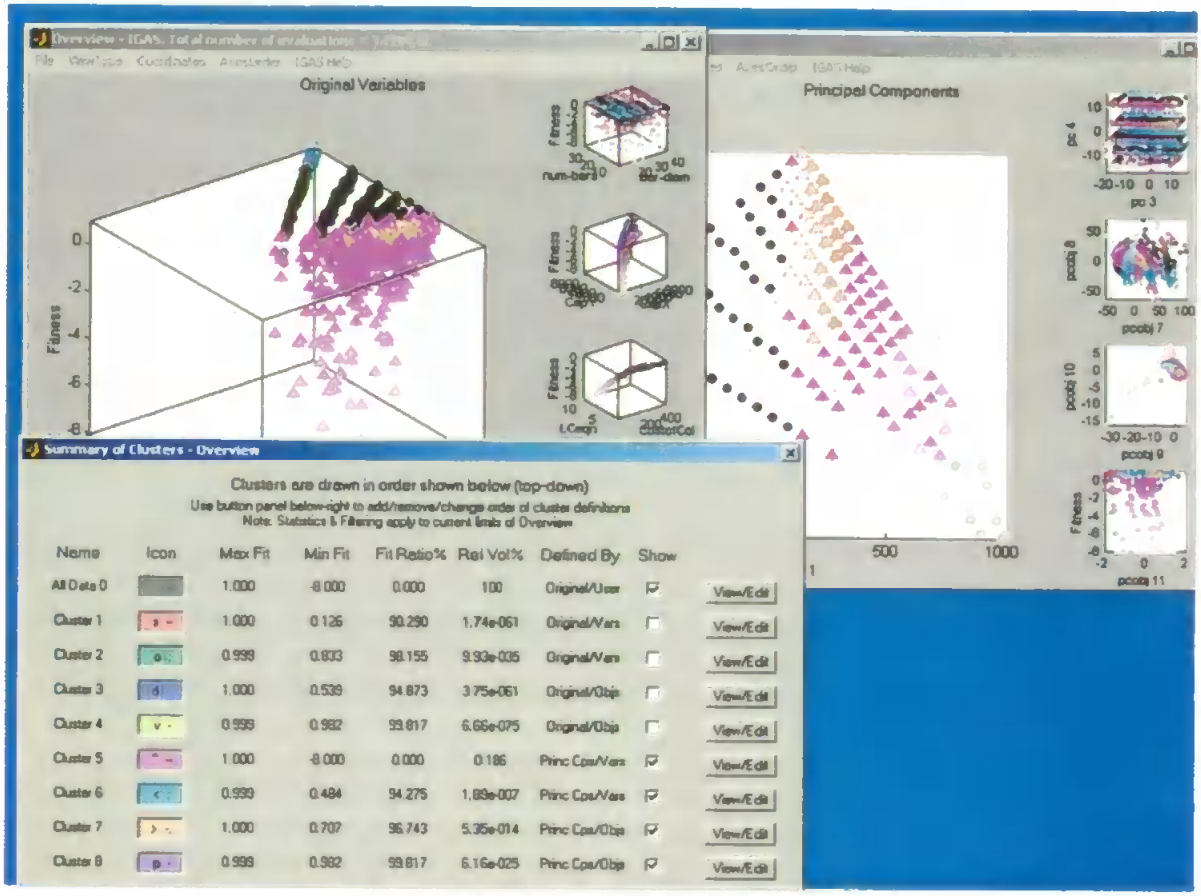


Figure 8.33: Clustering and Run GA available in variable space or objective space and principal component version of both spaces.

8.4.3 Optimising Column Cost and Evaluating Feasibility Robustness

In this function the main objective is to minimise the cost of building the column. The cost is a function of the cost of concrete, reinforcement steel and formwork, so is slightly different to the *%Area* objective used in the previous function; *%Area* was only concerned with the relative amount of steel used in each column. Because the column can have any size the cost is a fairer way to compare columns. Obviously the small columns will have very small cost but they violate the load contour equation constraint. Again there are alternative ways of tackling this problem, either constrain the search space in the system and try to force the GA into finding feasible solutions or change the objective function directly so that infeasible solutions are given low fitness. The second option is easier to visualise because good solutions are at the 'top' of the design space surface and the clustering algorithm will return the best clusters taking the constraint into account.

Figure 8.34 shows the result of optimising the cost of column but penalising infeasible solutions (*LCeqn*=1.02 is just feasible), the penalised solutions have low fitness so are of light colour. Clustering in variable space reveals the 'best' solutions are those of average size with few, large sized reinforcement bars (note the complete column is shown in the individual view). This reflects a trade-off between the amount of concrete and steel used that both contribute to the cost of the column, however the fact that all the highlighted designs use 40mm bar sizes is also an anomaly of the clustering algorithm working in discrete space. The algorithm splits the designs into diameter size, finds the best solution (in this case a column with 40mm bars), and thus returns a cluster with only 40mm bars. An engineer may have reasons for discounting the solutions with large, but fewer, bars, especially if a certain number of bars are needed to stop the column exploding.

The trade-off graph in objective space shows the amount of discontinuity present in this version (Figure 8.35 left). Automatic clustering in objective space again reveals the diversity of design options available (Figure 8.35 right), although the slightly different clustering results from Figure 8.34 indicate the complexity of the mapping between the two domains. It is worth bearing in mind that the clusters are computed using the 'dependent variables', not shown on these plots, which may discount some useful results. Further GA search 'inside' a manually created cluster in objective space shows the variety of solutions that could be used formed to make 'good' solutions (Figure 8.36); designs have been colour coded according to the reinforcement bar diameter.

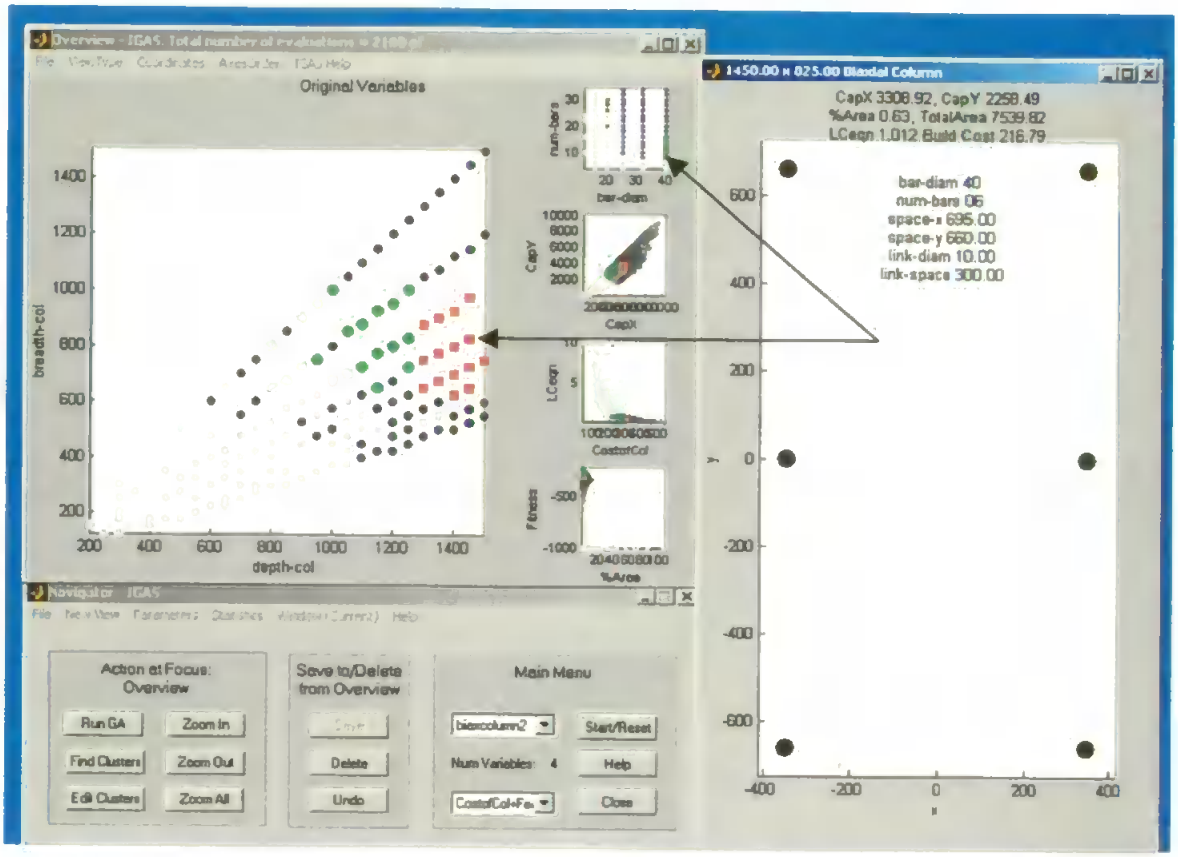


Figure 8.34: Optimising column cost, but penalising feasible solutions (with *LCeqn*>1.02). Clustering in variable space, the cheapest feasible solution is shown.

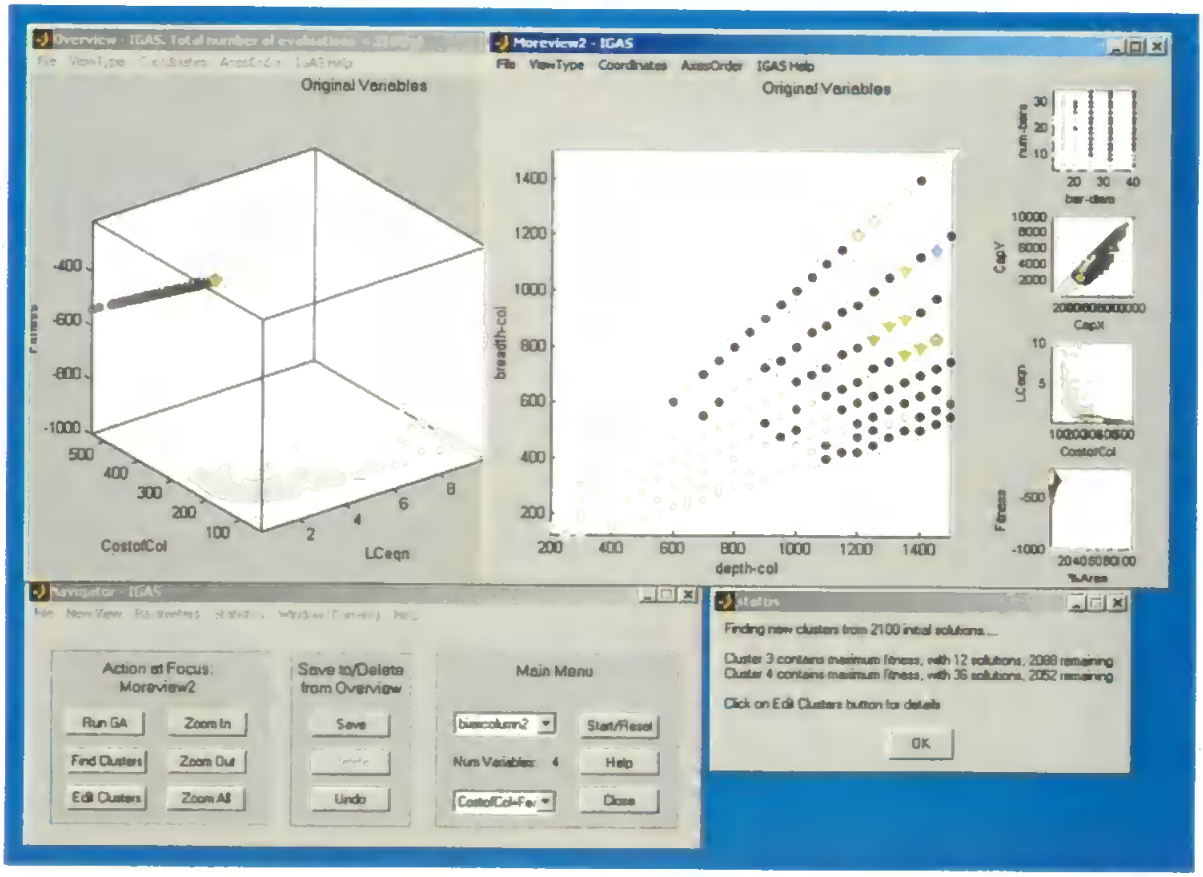


Figure 8.35: Clustering in objective space, note the discontinuous effect in the objective space trade-off graph (infeasible solutions have light colour).

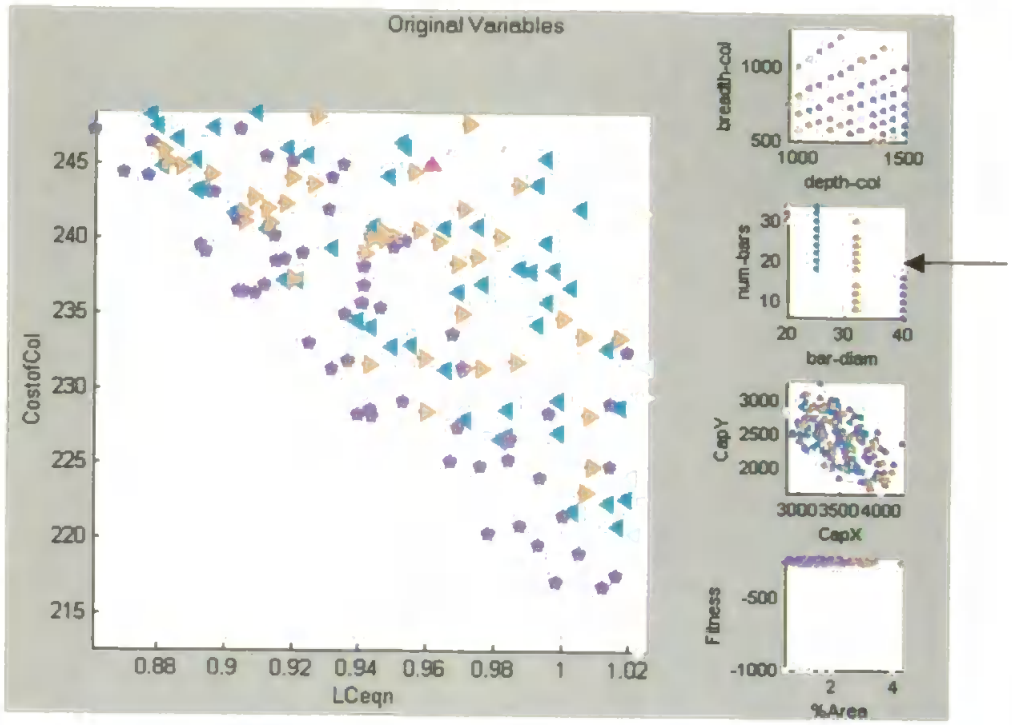


Figure 8.36: Further GA search in the required region of objective space. Solutions are coloured according to reinforcement bar size (see second plot down on right hand side).

Analysing the robustness of solutions is made more complex with the inclusion of the constraint. As well as evaluating the sensitivity of variables due to the change in cost of the solution, it is also necessary to ensure that changes in variables do not cause the solution to become infeasible. This form of investigation is known as "feasibility robustness" (Parkinson *et al.* 1993, Du & Chen 2000). The system can be used to assess robustness in the usual way using the filtering mechanism. Figure 8.37 shows the result of filtering the clusters found in variable space (Figure 8.34); very strict filtering is used (less than 1% of the fittest solutions are kept) with the result that any one solution remains in the red cluster. There are a small number of solutions remaining in the green cluster, all of fairly low cost, however some of the solutions are also infeasible. In fact a slight change in the column size of a near-optimal solution (Figure 8.37a) has resulted in an infeasible solution (Figure 8.37b). Looking at the constraint boundary on this figure shows that there are many solutions between the green solutions of different column sizes, so neighbouring solutions in variable space do not map to neighbouring solutions in objective space (as is

generally the case). It is easier to analyse robustness of solutions for this problem than the first biaxial column design problem, but the same difficulties with discrete inputs still arise.

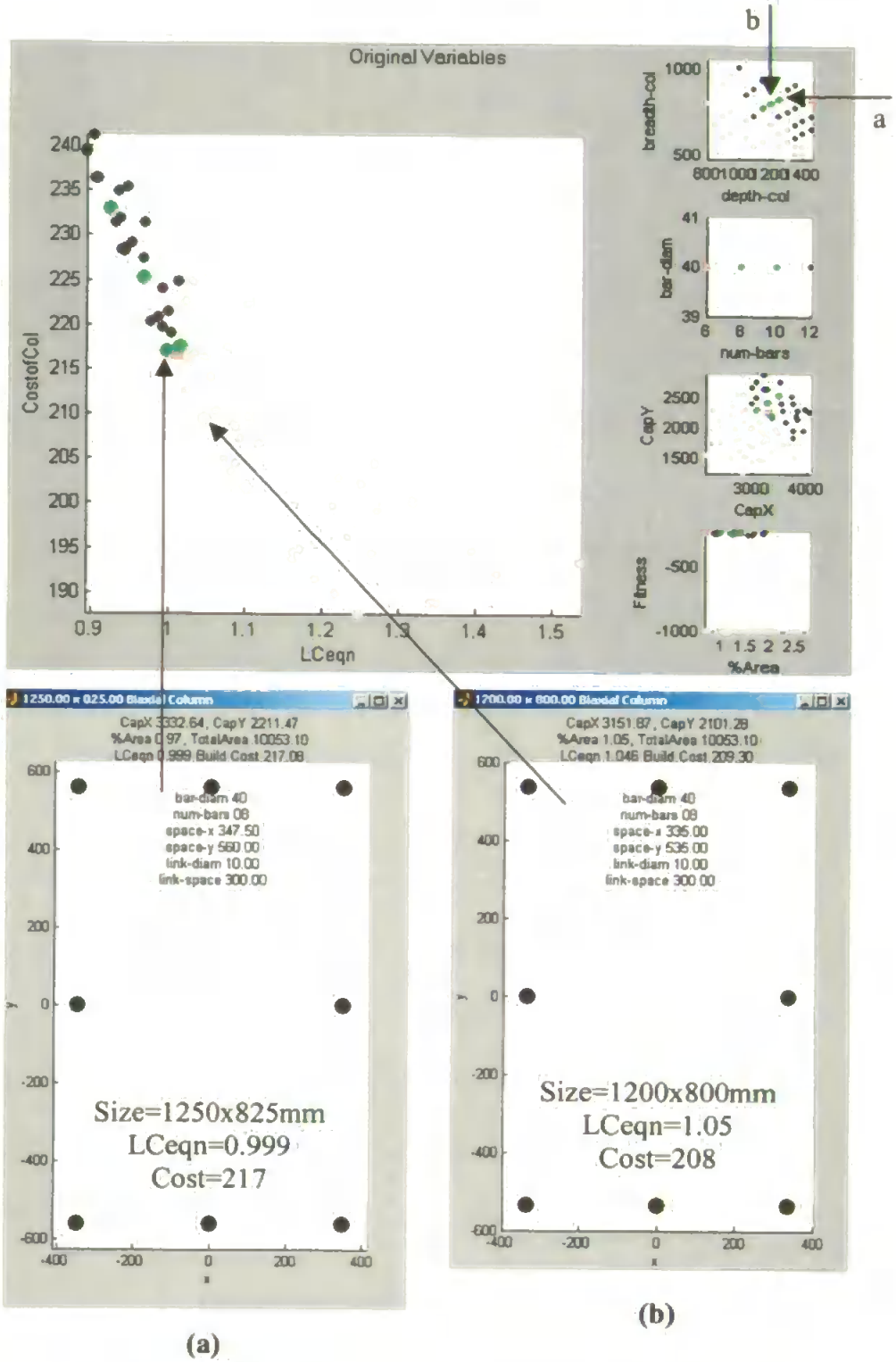


Figure 8.37: Filtering of clusters identified in variable space (Figure 8.34) to assess feasibility robustness. The pertinent objectives are displayed in the main plot. Green solutions are quite separated in objective space. A feasible solution shown (a), but a neighbouring solution in variable space is infeasible (b).

8.4.4 Expert Evaluation

As well as the contributor, two other civil engineering design experts from the University of Plymouth were invited to evaluate the system working on both biaxial column design problems. The author demonstrated the system and showed views of the search space as required while their comments were recorded. They were also asked to fill in an evaluation questionnaire at the end of the session, reproduced in Appendix E.

They were impressed by the variety of solutions that were displayed and compared. Again the ability to see the details of each solution was remarked on as particularly useful to get a feel for the designs being produced. They also liked the facility to zoom in on regions of objective space and generate further designs in the pertinent regions using the genetic algorithm. There was very positive feedback about the visualisation of solutions and the ability to see good and bad solutions together (Appendices E.2 and E.3).

A lot of discussion was provoked by such interaction with the system and looking at individual designs, for example those shown in Figure 8.38; the experts confirmed that they could make their own design decisions from this information. From this picture it is clear that there are a variety of solutions around $LCeqn \approx 1$ with different bar sizes. However the cheapest designs use a reinforcement bar size of 40mm; although these bars are larger, less of them are needed to ensure feasible solutions, thus reducing the cost (steel is the most expensive material – see Table 8.4). When the columns are made in practice, more bars are required to improve the "buildability" of the column and to stop the column exploding. The two intermediate designs (25 and 32mm) may be better design options for this, although they are more expensive to build. The best 20mm design (top) is by necessity large and has high cost, confirming that 20mm bars are not practical to support loads of this magnitude. Some concept of "buildability" may be included in the objective function by restricting the distance between bars, but much of the concept is subjective.

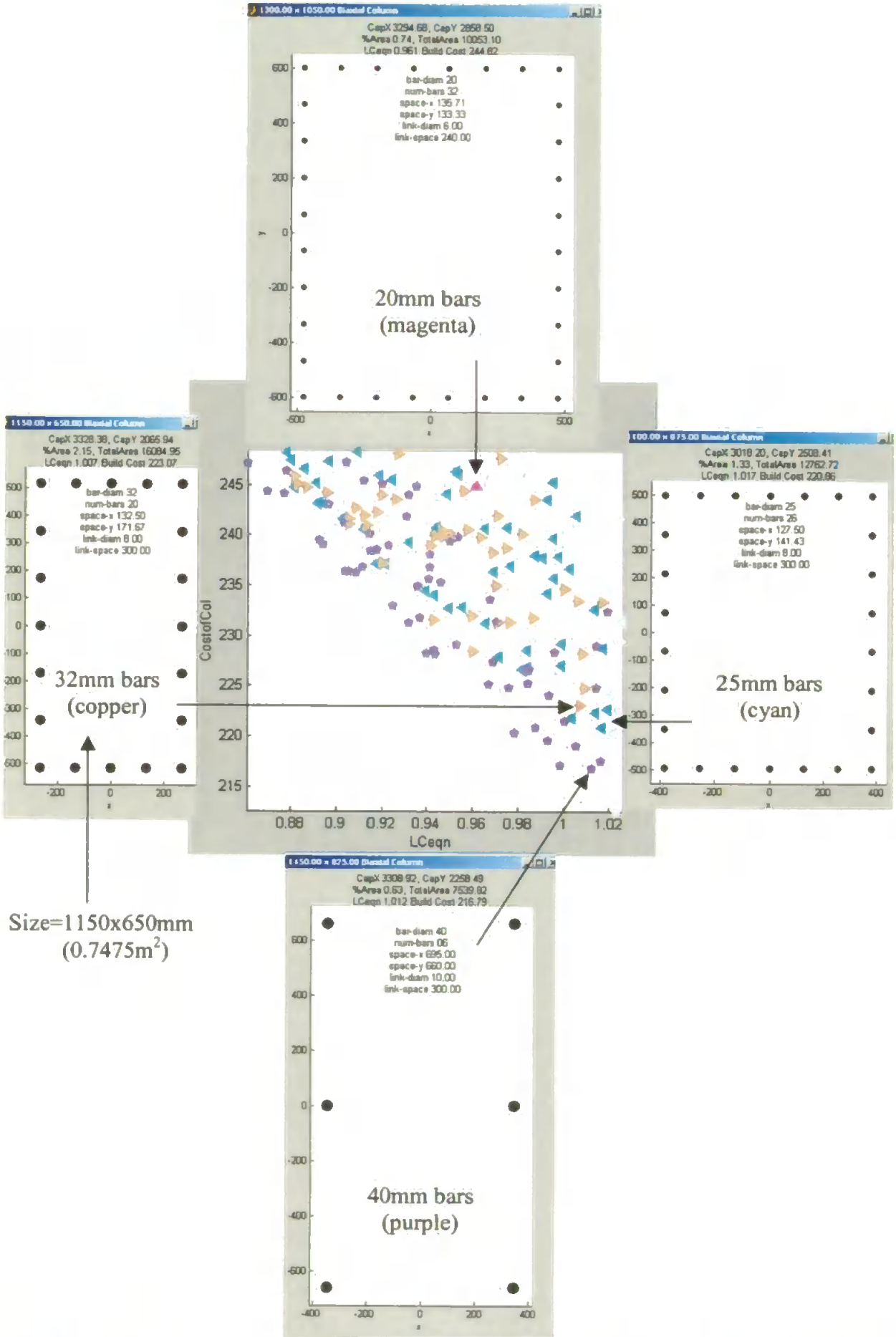


Figure 8.38: Alternative column details (sizes to scale - see Figure 8.36). Designs may be chosen for different reasons, for example "buildabilty" or materials available.

The experts confirmed that design decisions are made due to external causes that are dependent on the individual job in hand. For example the size of the column may be constrained by the size of slabs of formwork that has been supplied on site – to reduce wastage only certain sizes can be cut from the slab. There may be other constraints on the size of the column such as the size of cladding used, to ensure net floor space is maximised in a building or the dimensions of horizontal beams that the column should complement for aesthetic reasons. Also the diameter of available bars may be limited by the supplier due to ease of transport or manufacture – often smaller bars are cheaper to buy (presumably because there is less wastage involved during manufacture). All these scenarios are known to the engineer during the decision making process but are not present in the objective function, so more design options generated by the system will enable the subjective constraints to be met. If time for further analysis is available or a more detailed study is required, some of these constraints could be coded into the objective function, reducing but concentrating the design choices available.

Feedback about the robustness of solutions also confirmed that once a design has been chosen it will be built without much modification and is over-designed anyway for safety reasons, so the number and size of bars will not affect robustness nor will the exact location of the bars. In Appendix E.2 the evaluator reported that the system did not help to evaluate the robustness of solutions and commented: "As a practitioner I was making observations for myself". This is another example of domain knowledge being used in the decision making process; the expert used his experience to evaluate whether a solution was likely to withstand the load, knowing the likely variations and tolerances on design variables. However in further discussions this evaluator confirmed that the display such as that given in Figure 8.38 does help to understand the objective function and analyse the behaviour of similar column designs.

During the demonstration the concept of using the negative GA to 'solve' the engineering design problem was introduced. The feedback in Appendix E.3 (questions 3 and 6) indicates that the procedure was not understood and other functions in the system needed clarification. As documented the evaluation of robustness is not easy to demonstrate on this problem, so contributed to further confusion. Nevertheless both evaluators confirmed that the system showed that the second biaxial problem was robust (behaved as expected) and produced a number of designs that could be potential solutions to varying scenarios.

Following this feedback some alternative design scenarios were implemented in the system to investigate further potential of understanding the problem. It was suggested that different designs could be discovered if the cost of the materials used to build the column were to vary. For example in some countries steel is very difficult to obtain and thus the relative cost is much higher. This scenario can be simulated by halving the cost of concrete and leaving all the other input parameters unchanged. Figure 8.39 shows a similar trade-off picture to that given for the original scenario in Figure 8.38. The cost of all columns is obviously reduced, but optimal columns (in terms of cost) are a lot larger with more concrete used to fulfil the design constraint (compare size of the best columns with 32mm bars, details shown to the left of the figures). Columns with 20mm bars are more successful in this scenario, but a lot of small bars are needed to support this size of column, increasing the cost. However the best columns in terms of cost are again those containing a small number of 40mm bars. The amount of steel used is relatively low for these designs, but the larger bars have just enough moment affect in the corner of the column. Another reason why the best designs contain large bars may be due to the representation of the column in the GA; the depth of the column can be a certain fraction of its breadth so there are many more small column sizes available than large sizes (see *depth-col* against *bread-col* trade off in figures). In this scenario, where larger columns with less steel are

theoretically optimal, there are less design choices available. But for smaller columns with fewer (larger) bars there are more design choices, so these are more likely to be found by the GA. An alternative representation of the problem is advised for future investigation.

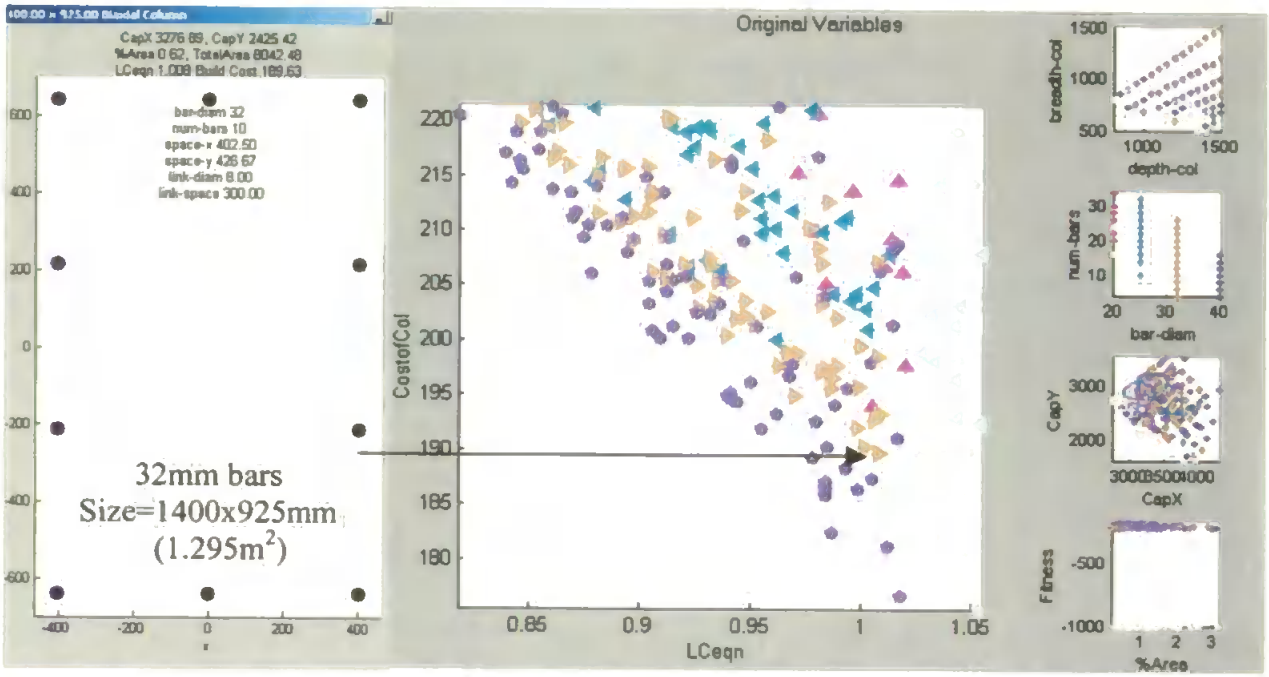


Figure 8.39: Changing design scenario: reduce concrete cost from 60 to 30. Result is bigger columns result with less reinforcement.

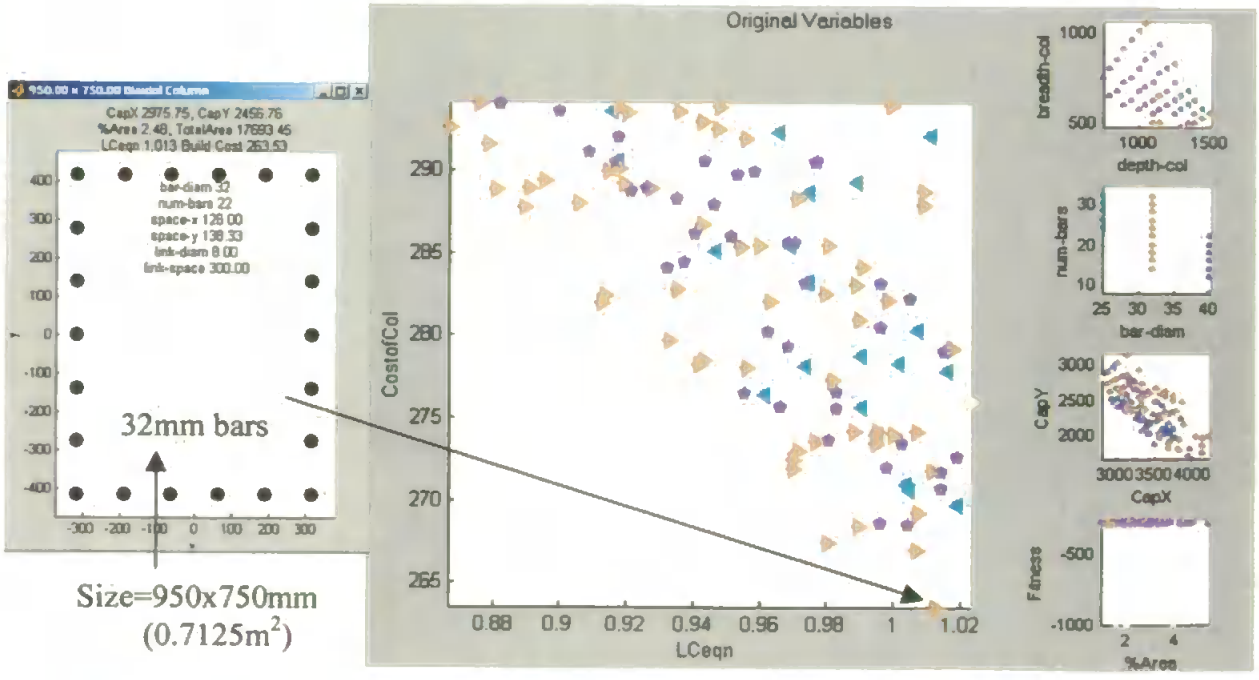


Figure 8.40: Changing design scenario: increase concrete cost to 120. Smaller columns with more reinforcement, no column with 20mm bars is available at all.

Conversely if the cost of concrete is increased relative to the original parameters, much smaller columns with more steel reinforcement are optimal (Figure 8.40). There is a complete absence of columns with 20mm bars amongst these good designs because there is not enough room in the small columns to contain the required number of small bars. In this case the columns with 32mm are optimal in terms of cost; they beat the columns containing 40mm bars because a change in the number of bars causes less change in fitness, so they have more chance of being an optimal design. However there may be other reasons for all these effects, including convergence to sub-optimal designs by the GA.

In summary the knowledge discovery aspects and potential for visualising more complicated engineering design problems most impressed the experts. The variety of design options are very useful for the engineer who may have other constraints to fulfil that are not explicitly expressed in the objective function. The evaluators also agreed that such a system was novel in engineering design and optimisation processes. They suggested the system would be even more beneficial in more complicated civil engineering design projects such as conceptual design of whole buildings or construction of football stadiums. For these projects even more design variables and uncertainty exists, so visualisation tools would improve understanding and increase the number of choices considered. If a better design is found the potential savings on costs and material used is considerable.

8.4.5 Conclusions

The second biaxial column design contained more domain knowledge and returned solutions that could be used in practical situations. Many design options could be obtained by concentrating search in the desired region of objective space, allowing the engineer to make choices between designs based on feasibility, cost and more subjective constraints such as "buildability" and ease of manufacture. The problem was more suited to theoretical evaluation of objective robustness due to fewer decision variables and a more continuous

search space. Some conclusions on the robustness of relative bar and column sizes could be drawn from the analysis. Optimising column cost whilst penalising solutions that violate the design constraint demonstrated that the system could be used to evaluate the feasibility robustness of solutions. It was shown that changes in the discrete variables caused relatively large changes in the objective space so that designs on the edge of the constraint boundary easily become unfeasible. Further analysis could be undertaken to assess the robustness of feasible designs away from the constraint boundary. However it was concluded that the representation of solutions was still too coarse grained to enable realistic analysis of robustness for this problem.

Feedback from expert evaluators confirmed that the system succeeds in its goal of presenting solutions to engineering design problems, provoking discussion and helping users understand the search space and confirm the knowledge of experienced engineers. The visualisation of individual solutions and comparison of solutions in relevant regions of objective space were particularly praised (Appendices E.2 and E.3). The system's ability to generate more good solutions in the regions also impressed. The visualisation of solutions in alternative coordinate systems was not presented to these practitioners, but a lot of discussion was generated from a small number of two and three dimensional plots and demonstration of simple interactions as shown in the this section and Section 8.3.

From the discussion and feedback it is clear that not all the displays and functions of the system were understood at first viewing (Appendix E.3), particularly the use of the negative GA and robustness evaluation. In reality columns are over-designed to ensure the specifications and safety requirements are met, so for individual designs the robustness issue has already been satisfied. For this problem the engineers evaluated robustness using their intuition and experience, this may be true in many engineering problems, particularly those with discrete variables. Nevertheless conclusions on the behaviour and robustness of

the objective function itself could still be drawn from assessing the relationship between neighbouring solutions. It was suggested that alternative design scenarios could be demonstrated with the system, further simulation confirmed the varying emphasis on column sizes and amount of reinforcement needed to satisfy different environmental and financial conditions. From the discussions it became clear the evaluators used their knowledge and experience in the decision making process, supported by the visual information rather than driven by it.

8.5 Overall Conclusions

The case studies described in this chapter reveal the diverse nature of real world applications and the consequent difficulties that need to be overcome by an interactive engineering design system. The rainfall runoff case study exhibited continuous decision variables and a single objective that was related to minimising an error function. This function was most similar to the test functions presented in Chapters 6 and 7, responding well to the clustering algorithm and the use of negative GA search. In contrast the two biaxial column problems had discrete inputs and multiple objectives exhibiting discontinuities because of hard constraints; the automatic clustering and evaluation of robustness procedures were less successful here. A large amount of domain knowledge was encoded into the latter functions, although choosing solutions that solve the problem depends on the particular design scenario and subjective constraints known only to experienced engineers. The system helped to support the decision making process by providing access to multiple design options. The biaxial column design problems are really detailed design scenarios, however the notion of moving objects around discrete space is not very different to whole building design. The need to make choices in a multiobjective, subjective search space is also comparable to such a conceptual design problem.

The visualisation of the search space helped the experienced contributors confirm the behaviour of their problems and suggested new solutions. For the rainfall runoff problem removing redundant input parameters was suggested by the visualisation system and, after implementation, new feasible solutions with similar fitness values were found outside the original search limits. These actions essentially changed the model, but they revealed how the input parameters to the problem interact. For the biaxial column design problems the system needed to be modified to allow clustering in objective space and force the GA to search inside defined regions of objective space by penalising the fitness of solutions found outside. This method generated solutions at the constraint boundary in objective space, provoking a lot of discussion amongst engineers. Some new design options were found that improved on previously published work in terms of individual objectives.

The true strength of the system is its ability to provide a number of different designs allowing the experts to choose between them using their knowledge of the design situation. The representation of individual solutions was seen as particularly useful to support this knowledge, providing immediate visual feedback of a prototype design instead of a set of figures that are difficult to decipher. Visualisation of individual solutions helped the engineers choose a "buildable" design in the biaxial column problem, whilst in the rainfall runoff model comparisons between the predicted and observed solutions could be made and areas of mismatch identified.

Using the system to evaluate robustness was successful for the continuous model but less so where discrete inputs are involved. The rainfall runoff model proved to be very sensitive to input variables, the negative GA could be used to analyse the robustness of similar sized regions and determine the more robust parameters to some extent, but the overall conclusion is that the model itself was not sufficient to model the complicated

relationship. The contributor knew this lack of robustness of the model, but the tools on the system confirmed the facts and showed that the robustness of future models could be assessed in the same way. For the first biaxial column problem the low level representation of the decision variables resulted in repeated and redundant solutions, so that determining a relationship between individuals was very difficult to determine. Selecting regions of variable space and using a negative GA was not practical in this case. However for the second biaxial design problem fewer variables were used that responded better to clustering and theoretical robustness evaluation; in this case feasibility robustness could also be evaluated. Use of the negative GA gave some indication as to which designs are robust, but even for small clusters an infeasible solution is not far away. This analysis showed that the relationship between variables and objectives is very complicated. In practice an individual solution will be over-designed so robustness is rarely an issue, but the analysis showed the potential of the system to investigate the robustness of feasible solutions. When viewing the system, engineering experts also confirmed that both biaxial column problems were robust.

Feedback from expert evaluators was very positive, most views and processes were generally understood, but it takes time to explain some of the concepts of the system and a significant amount of time would be required to train people in using the system to its full potential. Use of the negative GA in the discrete problems was not always understood and further work is needed to clarify the potential and applicability of the idea in such domains. Clustering in alternative coordinate systems has the potential for providing a lot more information about the natural clusters in a data set or problem. However in future work a clear understanding of the results returned is necessary so that users have the incentive to learn and confidence to view and manipulate the data in these alternative ways.

For this and many problems the clustering technique sometimes returns clusters that are too small to be of use or interest. Although they do contain the 'best' solution, the rest of the information is not useful. Clustering in objective space is sometimes useful but usually more informative results were found by manual clustering. This phenomenon is closely linked to the curse of dimensionality; as the number of dimensions increases less solutions are 'neighbouring' each other. One solution is to let the user choose which variables or objectives are used in the clustering algorithm, so that irrelevant dependent variables and discrete variables that cause problems can be ignored. A more flexible clustering tool would also allow controlled evaluation of robustness by constraining decision variables (similar to the approach of Tweedie *et al.* (1996b)) whilst freeing the objectives to discover the worst case situation.

Another avenue for future work is to incorporate Pareto ranking and optimisation into the system, providing further design options at the trade-off boundary between objectives and constraints instead of specific regions found using the weighted sum approach. The system would then generate solutions even more relevant to the design problem whilst allowing an engineer to further guide the search, filter the results and make choices between those designs, as advocated by Parmee *et al.* (2000) and Rafiq (2000).

These case studies have also revealed some major differences between theoretical concepts of engineering design that the system was intended to solve and the methods used by practicing engineers in real world problems. Theories to evaluate robustness are usually based on the assumption that the problem contains continuous variables and the function is well behaved with many loose design parameters. Some problems like this do exist, so the statistics and worst case evaluation as implemented in the system should be retained. However the civil engineering design problems presented in this chapter contained a lot of domain knowledge so the function is guided towards the desired region of the search space

and the need for robustness evaluation is virtually eliminated. Discrete decision variables and objective functions with discontinuities caused by constraints are much more difficult to visualise and analyse. Evaluation of robustness and feasibility robustness seems to be performed internally by the engineer, although visualisation can help to confirm the intuitive answer. Furthermore the subjective constraints that were not coded into the objective functions allow many different scenarios to be considered by changing a few input parameters. In the cases where the engineer has so much knowledge of the problem, it is less important to provide sophisticated clustering and robustness evaluation tools, but more important to allow visualisation of multiple results. Rather than reacting to new information and being guided by the system, practical engineers would use the system to confirm design solutions and could explain the reasoning behind the choices to less experienced engineers.

Chapter 9: Conclusions

9.1 Outcome of Research

9.1.1 Summary of System Developed

A visualisation system designed to help engineers solve engineering design tasks using evolutionary computing was created following extensive research into the requirements for such a system (Chapter 2) and the tools needed to achieve those requirements (Chapter 3). An innovative clustering algorithm based on kernel density estimation (KDE) was introduced to quickly identify the main clusters pertinent to engineering design; clustering in alternative coordinate systems such as the principal or independent components is also possible, often revealing the natural partitions in the data (Chapter 4). Standard multivariate visualisation techniques that do not distort the data were incorporated into the system (Wong & Bergeron 1997, Spence 2003) and colour is used to highlight clusters defined by the user or clustering algorithm (Section 5.2). To encourage user interaction a flexible, easy to use interface was introduced that presents information in an understandable way (Section 5.3). Users have the opportunity to define their own clusters, change the definition of previously defined clusters and view useful statistical measures. The interface allows the user to generate data and search for clusters inside or outside previously defined regions (Section 5.5). Clusters defined in alternative coordinate systems can easily be related to the original design variables.

It was found that a lot of information could be returned using a simple genetic algorithm (GA) with a low number of generations. Diversity was encouraged by mutating duplicated individuals (Section 5.4) in the binary string; this scheme will remain effective providing the number of generations and string length are relatively low. Robustness of regions of the data can be confirmed by an inventive technique of filtering the data in terms of objective value and using 'negative' GA search (Section 5.6).

Following the introduction of real world problems, the system was modified to take into account multiple objectives and discrete variables. The clustering algorithm was extended for use in objective space; new solutions can be generated in regions of objective space by penalising solutions that fall outside the required region (Section 8.3.2). This enabled a designer to conduct a concentrated search in a particular region of interest, creating more knowledge and helping the decision making process. Individual details of designs generated by the system can also be viewed separately, allowing direct comparison between the different artefacts.

The system and clustering algorithm were originally developed for continuous problems and tested using continuous test functions; they were then tested on real-world, discrete problems with mixed results (as summarised in the following sub-sections). For the approach to fully exploit the advantages of evolutionary computing for the majority of engineering design problems, further development and testing is needed to take into account discrete variables and combinatorial functions. Suggestions on how to proceed in this direction are given in Section 9.2.

9.1.2 User Evaluation Experiments, Results and Conclusions

To evaluate whether the system achieves the philosophy of promoting collaboration between the human and computer (thus helping to understand engineering design problems), the system was initially tested using novice subjects with varying experience of engineering design. This preliminary evaluation was designed to assess the usefulness and applicability of as many features of the system as possible without the need to train experienced engineers in how to use the system. Unique test functions were designed to simulate engineering design problems; unevenly distributed optima with varying height, width and amount of noise present (Section 6.2). The participants were asked to complete

an engineering design task that required them to locate robust regions of the search space and evaluate the relative quality of each defined region (Section 6.3). A region was defined as robust if the fitness of solutions found inside the region were above a minimum tolerance level of the local optimum. The quality of the region was defined as the product of the size of that region in variable space (hypervolume) and the fitness value of the local optimum. Statistical measures were introduced to compare the accuracy of the user-defined regions to theoretically 'ideal' regions (Section 6.4). Another set of statistical measures analysed the sampling of the ideal regions by the user-generated data, this second set of metrics were compared with results from three benchmark multimodal genetic algorithms: the simple GA, GA with sharing and GA with deterministic crowding (Section 6.6).

It was impossible to make any firm conclusions from the statistical results due to the small number of participants used, but analysis of individual results revealed the differences in behaviour of the users due to their level of understanding of the system and the problem (Section 7.2). Individual algorithms achieved better results than the users on specific measures; this confirmed the behaviour of those algorithms and their usefulness to solve specific tasks (Section 7.4). Nevertheless none of the algorithms outperformed the users on all measures for the complete engineering design task. It was shown that even a devoted hybrid algorithm would need help to solve the task, so the need for incorporating a human user into the exploration and design process was once again confirmed.

Feedback from the participants after the experiments confirmed that they found the system useful and enjoyed the flexibility of interacting with the data (Section 7.3 and Appendix D). The ability to generate and view clusters of the data using colour and confirm the robustness of those regions of the search space using a 'negative' GA were seen as particularly beneficial. The feedback suggested certain features of the system that could be changed and improved (Section 7.5.1), especially combining processes to remove

repetitive actions such as to check the robustness of regions (Section 7.6.1) and automatically find robust regions (Section 7.6.2 - this is similar to the sharing-based sequential niche technique of Beasley *et al.* (1993)). Clustering in alternative coordinate systems was not used by any of the evaluators; brief experiments on the test functions were inconclusive as to the usefulness of the technique (Section 7.6.3), but further investigation is warranted (see Section 9.2).

The main objective of the research was that the system should fulfil a dual requirement (Section 1.2). Firstly, the computer should generate data and identify robust, high quality regions of the search space. Secondly, the user should be allowed to interact with the data and guide the search as required. Both system requirements appeared to have been achieved, however further examination of the users' results revealed that some regions defined as high quality in the engineering design task were not identified by the user or made obvious by the system. So in some cases the system failed to identify potentially high performing regions.

A critical analysis of the user experiments highlighted three main factors that caused the participants to miss important data and fail to identify robust regions. The first reason for this failure is that the clustering tool did not look for relatively low fitness regions that could potentially be robust and of high quality (Section 7.5.2). This suggests that the clustering tool needs to be modified to allow the user to specify the range and type of data required and possibly more sophisticated clustering techniques are needed to locate this information. However, the fact that the users did not realise that they should be looking for such low fitness information suggests further reasons for failure: they did not fully understand the engineering design task and the definition of robustness.

The engineering design task was very ambitious and asked a lot of novice users working on an unfamiliar system (Section 7.5.3). The participants all had varying theoretical knowledge of engineering design; in fact some of those with less knowledge had more success because they considered the task as visual problem solving. This fact suggests that the system could also have further application in cognitive analysis and problem solving. For further testing of the system as an engineering design tool it was recommended that specific features are formally tested by giving the users simpler tasks and slowly building up their knowledge of the system until more complicated problems can be tried. More realistic problems that involve discrete variables, multiple objectives and even real world problems would also be advantageous.

The engineering design task and definition of robustness were designed to encourage decision making, from the feedback it is clear that the system helped to support this. However, the critical analysis suggested that the definition of robustness was incomplete because the tolerance level was set relative to the objective (or fitness) value of the local optimum. Users needed to make subjective comparisons between regions using the statistics provided on the interface (Section 7.5.4). An alternative technique for evaluating the robustness of regions of the search space was suggested by setting the tolerance level in variable space and comparing the absolute values of maximum and minimum objective (or fitness) found in the region (Section 7.6.4). This simpler method removes a lot of the uncertainty and is more directly related to real world design problems where tolerances are set due to limitations on materials and target specifications according to Taguchi (1986) and Tweedie *et al.* (1996b). So the experiments and analysis facilitated understanding of the theoretical evaluation of robustness.

9.1.3 Case Studies: Results and Conclusions

After the evaluation of the system working on artificial test functions (using theoretical concepts of engineering design and robustness), its usefulness and effectiveness was tested on real world problems. The system was shown to engineering design experts and demonstrated working on problems they designed or were very familiar with.

The first case study entailed finding parameters to model daily river flow using the history of recent rainfall and river flow only (Section 8.2). The problem had continuous variables and a single objective related to minimising the error function between the model and observed values; this function was the most similar to the test functions employed in the user experiments. The contributor was able to confirm some attributes of the model through the visualisations. The noisiness of the model was apparent due to the variety of parameter configurations that returned similar values. It could be seen that some factors that relate to older days rainfall were very small and could be discarded from the model. More good results were discovered by widening the range of parameters in this revised model, confirming the relationship between some of the parameters in the model (Section 8.2.2). However, the use of negative GA search revealed the lack of robustness of these results and thus the lack of robustness of the model; this suggests other parameters may be required, such as those that describe the geology and vegetation of the catchment area. Additionally visualisation of individual results indicated where the model differs from the required output; these parts of the data may require a different model to simulate the data, suggesting that multiobjective optimisation could be applied to the problem. Traditionally rainfall runoff models are tested on multiple data sets with the aim of finding the optimal number of parameters to achieve the most accurate and consistent results (Davidson *et al.* 2000). This study indicates that the system could be used to evaluate more sophisticated models and speed up the validation process.

The other two case studies were forms of the biaxial column problem that exhibited very different characteristics from the first study: multiple objectives and discrete decision variables. For the first biaxial column problem the decision variables were the diameter and position of each reinforcement bar in the column, there were two main conflicting objectives: to ensure the column would withstand the applied load and to minimise the amount of steel used in the column (Section 8.3.1). It was virtually impossible to visualise all the decision variables and make conclusions from the clustering analysis in this discrete domain (Section 8.3.2). However clustering in the required regions of objective space and generating new solutions here produced new designs that used less steel than those published by the contributor (Rafiq & Southcombe 1998), although the feasibility of these designs needed verifying (Section 8.3.3). Principal component analysis of the objective space appeared to highlight solutions along the trade-off curve or Pareto front, indicating a possible application of visualising and clustering in alternative coordinate systems. The robustness of this model was confirmed by viewing the change in construction of individual designs, however the robustness of individual solutions was impossible to evaluate due to the discrete nature of the diverse inputs. The real advantage of the system was seen in its ability to produce a number of different designs in the pertinent region of objective space that an experienced engineer could modify, taking advantage of domain knowledge (Section 8.3.4).

The second biaxial column problem contained a lot more domain knowledge than the first; the number of variables was reduced because the number and diameter of the bars could describe the configuration of the whole column (Section 8.4.1), although the depth and breadth of the column were allowed to vary. The system could be used to theoretically evaluate robustness of solutions (Section 8.4.2). These decision variables were also discrete, so only general conclusions could be drawn about the most robust configuration. Evaluating robustness in alternative coordinate systems was also possible but further

understanding of the results is required. For this problem one of the objectives was the cost of the column, whilst the solutions were considered infeasible if they did not satisfy the design constraint of resisting the required load and biaxial bending moments. So analysis of the "feasibility robustness" of solutions could be made (Parkinson *et al.* 1993, Du & Chen 2000). Again the discreteness of the variables caused the solutions to easily become infeasible, but interesting relationships between the variables and objectives were revealed (Section 8.4.3). Further extensions to the clustering tool and system were suggested for application in real world domains: as well as allowing the user to choose which variables and objectives contribute to the clustering analysis, it should be possible to specify known constraints and tolerance ranges for the specific problem.

The system working on the biaxial column problems was shown to civil engineering design experts. The feedback was very positive indicating that visualisation of the search space and the ability to focus and generate new data as required was extremely useful (Appendix E). It is clear that some parts of the system were not understood and some training would be required for users to become comfortable with the system; the evaluation of robustness procedure caused particular confusion. For this model, however, experienced engineers over-design the column to ensure safety and to make it easy to build; so individual robustness evaluation has already been undertaken (Section 8.4.4). Further analysis of the problem revealed that the system could be used to generate many solutions to different design scenarios, confirming the robustness of the model and usefulness of the system.

The ability to visualise and compare individual designs was also seen as very informative by the evaluators (Appendix E.1 and E.2). This shows the importance of relating abstract designs (in terms of mathematical numbers and equations) to the physical artefact that will be manufactured; this is similar to the importance of creating prototypes

during software development (Boehm 1988, Mumford 2003 p.162). A drawing or computer image can inform designing (Schön & Wiggins 1992) but it should be remembered that the artefact is a context dependent representation of the design (Bucciarelli 1988) and the final design may differ somewhat to the drawing.

9.1.4 Overall Conclusions: Involve the Human and Computer in Engineering Design

From the discussion and conclusions of Chapters 7 and 8 it is clear that there is a marked difference between theoretical and practical concepts of engineering design. Solving the engineering design task in the evaluation experiments required subjective comparison between regions of the search space. Although the definition of robustness was explicitly given and easy to evaluate using the tools on the interface, the decision making process was based on ill-defined guidelines that were not related to a physical problem so were difficult to resolve. The system was designed to help in the conceptual stage of engineering design, but this amount of uncertainty is unlikely to occur in practice. The real world case studies demonstrated the improved results of increasing domain knowledge in the objective functions and the modifications to the system needed to exploit the results. The first case study contained continuous variables allowing the theoretical evaluation of robustness, such as that used in Taguchi analysis (Phadke 1989). In the final case study the evaluation of robustness and feasibility of solutions was performed internally by the engineers, so it was not considered necessary to assess robustness with the system. This was partly due to the discreteness of the design variables but also due to the subjective knowledge of the engineers – based on experience and solid facts. For these experts the visual information alone is enough to confirm good solutions and explain the design process to less experienced engineers.

The final case study was in fact a detailed design problem, this observation leads to the question: is the system capable of supporting the independent stages of design –

conceptual, embodiment and detailed design (French 1999)? This question is answered by considering the various design methodologies introduced in Section 2.2.2. Visualisation of the genetic algorithm output allows many combination of parameter settings and their effect on objective space to be assessed, supporting the systematic approach to engineering design where the required variables have been determined (Jones 1992, p. 104). The generation of a large number of solutions promotes discussion and encourages group decision making, thus the ethnographic process of design is also supported (Bucciarelli 1988, Baird *et al.* 2000). Visualisation of alternative artefacts could lead to creativity by 'moving' and 'seeing' new designs using reflective practice (Bucciarelli 1984, Schön & Wiggins 1992). In the current system domain knowledge is encoded directly into the objective function, but engineers may need to use subjective knowledge to identify truly feasible designs. Indeed the ability to generate new designs at the boundary of either explicit or implicit constraints is one of the major advantages of this system. The interactive capabilities of the approach would also be invaluable as a teaching tool, enabling less experienced engineers to analyse the interaction between parameters in a real world problem and immediately visualise the effect of those changes on the artefact. So systematic, reflective and ethnographic design methodologies can all be accommodated using the proposed system or a similar interactive approach, providing the problem can be formulated in a mathematical way. These methodologies are all used, to varying degrees, during the different stages of engineering design, so such an approach could be adapted to facilitate the complete design process.

The system was designed to promote collaboration between the human and computer; feedback from the case studies suggests the proposed approach was successful. The importance of incorporating the human into a design system (Jo 1998) has been underlined by this work. Human input is particularly relevant between stages of the design process where risks need to be evaluated and goals possibly modified, mirroring the spiral

model of software development proposed by Boehm (1988). Nevertheless the system needs a lot more features to cope with the various stages of design and provide a smooth transition between them; recommendations given in next the section suggest how this can be achieved.

The proposed system could be used as a tool in the modern practice of multidisciplinary design activity (Jagodzinski *et al.* 2000, Mumford 2003). It is possible that analysing the interaction processes within the system and between designers could provide hints on how design is undertaken and some of these processes could be saved in a database to aid a connectionist model of design (Coyne 1990) or automated and added to an artificial intelligence model of design (Maher 1990, Gero & Maher 1993, see Section 2.2.3). However the complexity of each design problem is so large that computers can play only a supporting role for the foreseeable future. Thus during the development of systems for design, emphasis should be placed on the users and the different perspectives of the users (Bucciarelli 1988, Norman 1998, Baird *et al.* 2000, Mumford 2003).

9.2 Future Research Directions

The system supports some of the recommendations given by Parmee *et al.* (2000) for the implementation of an interactive engineering design system using evolutionary computing, however further recommendations such as the easy addition and removal of objectives and constraints would permit an even more flexible system, increasing the likelihood of finding "innovative or even creative solutions" (*ibid.* p. 198). These recommendations could be practically implemented in the system by choosing which objectives contribute to the overall fitness (and how they are combined) and avoiding the generation of data outside the constraint boundaries. Such a mechanism would allow the user to input domain knowledge without having to change the objective function code. The clustering procedure could also be modified to include specific variables or objectives, removing the problems that discrete

variables and unimportant dependent variables bring to the results; the evaluation of robustness and feasibility robustness within those regions would also be improved.

To avoid the problem of the user missing important information in the data, the clustering technique based on kernel density estimation (KDE) should be extended to take into account domain knowledge such as limits on variable values (tolerances) or objective values (constraints). The algorithm could be modified to bring other attributes of the data to the users' attention, such as regions of highest density or skewed data. However this KDE-based technique is not sophisticated enough to analyse complicated or non-linear data. Statistical and clustering techniques described in Sections 3.4 and 3.5 could all be adapted to find clusters in engineering design data and assessed for performance and usability (see Section 7.7). For a large number of parameters, it would also be useful to automatically identify the most sensitive variables or objectives (Packham & Parmee 2000).

Creating clusters in alternative coordinates systems sometimes revealed interesting characteristics of the data, but the applicability of defining such regions and evaluating their robustness needs more investigation. As well as undertaking further work to understand the true meaning of the results, it is also necessary to convince engineers of their usefulness. Engineers will have the incentive to understand and use the technique if it can be shown to assist real world problems.

The simple GA with mutation of duplicated individuals may not be sufficient for more complex problems that require a high number of generations or very long string length; the number of comparisons required could become unfeasibly large. As many engineering problems are combinatorial or contain primarily discrete variables, modifications to the genetic algorithm, clustering algorithm and interface would be

required, depending on the individual problem. It would be very difficult to visualise the variable space of combinatorial problems such as the travelling salesperson (Louis & Tang 1991) and design of water networks (de Schaetzen *et al.* 2000), as was discovered in the first biaxial column problem. In such cases it may only be possible to compare the fitness of solutions, however visualising the artefact (individual details of solutions) would enable the user to make subjective choices. A further extension would allow interaction with the artefact to constrain the search space, similar to how Louis & Tang (1991) encouraged users to influence further search by manipulating configurations. During the multiobjective case studies it was very instructive to perform clustering in objective space and visualise individual solutions; this procedure may be the most advantageous for similar problems.

More advanced genetic algorithms could be incorporated into the system, or given as options to the user, to speed up the process of finding robust local optima. For example multimodal algorithms mentioned in Sections 2.3.3 and 6.6 or techniques that attempt to find a smooth mapping between genotypic and phenotypic space (Harvey & Thompson 1997). These neutral mappings simulate redundancy in natural evolution (Shipman *et al.* 2000). As mentioned in Section 8.3.4 the system should be extended to allow the visualisation of multiple runs from different objectives (Parmee *et al.* 2001) and multiobjective optimisation using Pareto ranking. Pareto ranking produces a large number of design options along the trade-off boundary between objectives (Rafiq 2000), nevertheless even this approach needs human interaction and knowledge to determine the best designs.

The future potential of the system was indicated by the user evaluation results, however individual attributes of the interface and engineering design task to test the interface need improving. Some interface features were not immediately obvious and failed to bridge the gap between the psychological understanding of the problem and the physical

processes of the system (Norman 1986, 1998), additionally a clear understanding of the problem to be tackled is required to enable this gap to be bridged. More realistic functions should be introduced to simulate an engineering design task, such as combinatorial problems and functions with discrete variables, multiple objectives and discontinuity or constraints in the objectives. Initially the tasks should be simple and short, increasing in complexity as the users become comfortable with the system over a number of trials. Analysis of this problem solving would be useful for alternative applications such as cognitive psychology (Eysenck & Keane 2000). The ability to record the users' actions should be exploited so the system can learn how the user is designing and making decisions (Noy & Schroeder 2001). If the system can capture the design process and suggest actions that result in solutions outside the original search space then, according to Gero (1990), it can truly be called innovative.

Encouragement from knowledgeable engineers suggests the system should be employed on as many engineering projects as possible, conversely exposure of the system to the engineering design community would suggest further modifications that can only improve the system. The discussion generated in the case studies showed that the system would be particularly useful in a multidisciplinary conceptual design environment (Jagodzinski et al 2000). If the system is to be applied to a particular problem, where an in-house system may already be in place, a complete redesign of the system is needed so that interaction, visualisation of artefact and incorporation of evolutionary computing can be accommodated. In such a case the methodology of Mumford (1995) would advocate participative design so that potential users are part of the design process. The role of a "facilitator" is crucial to this design process, ensuring that all interested parties are included in the project and contrasting views can be discussed in a constructive atmosphere (Mumford 2003).

9.3 Summary of Contributions and Conclusions

9.3.1 Contributions to Knowledge

1. The System

The development of an interactive system that combines the research areas of evolutionary computing, engineering design and multivariate visualisation, featuring:

- A unique, flexible interface that allows the generation of data with an enhanced genetic algorithm - inside and outside pre-defined regions of the search space
- Parallel coordinate and scatter plot views (2D or 3D) to visualise any parameter combination of undistorted data, with a link to the relevant artefact

2. The Clustering Algorithm

Introduction of a novel clustering algorithm based on kernel density estimation that quickly identifies regions of the search space relevant to engineering design

- Clustering was also implemented in multiobjective problems; new data can be generated in pertinent regions of objective space using penalty functions
- The possibility of clustering and running a genetic algorithm in alternative coordinate systems (such as the principal components) was demonstrated

3. Negative GA Search

Novel use of 'negative' genetic algorithm search to find the worst case scenario to evaluate robustness and feasibility robustness

- Combined with a filtering mechanism, this technique can be used to define more robust regions

4. *Test Functions and Metrics*

The creation of continuous multidimensional test functions and metrics to evaluate the success of people and algorithms working on simulated engineering design scenarios

9.3.2 Conclusions due to Analysis and Implementation of the Approach

1. Decision making and knowledge discovery are supported by the many features of the interface, such as: multiple views, ability to visualise any combination of parameters, generation of data in arbitrary regions of the search space using the genetic algorithm, ability to automatically or manually define clusters in the data
2. The use of colour provides an explicit definition of significant clusters and helps to maintain consistency between different views and coordinate systems
3. Direct manipulation and the use of penalty functions in variable and objective spaces successfully enabled the generation of new solutions
4. Empirical evaluation could not statistically prove the advantage of interaction over multimodal genetic algorithms, but the algorithms need external help to solve tasks
5. Evaluating the robustness of regions and their automatic redefinition to ensure they contain good solutions was successful in continuous domains
6. Clustering analysis and evaluation of robustness in discrete variables and alternative coordinate systems needs refining and requires further investigation
7. The ability to visualise individual design solutions improves practical understanding of the problem
8. Comparison of solutions promotes understanding of the interaction between parameters in artificial and real world problems
9. Experienced engineers use subjective knowledge of constraints and robustness to solve problems and consider alternative design scenarios; they can use the visualisations to support a mental model of the design process
10. The visualisation and interaction capabilities of the system would be ideal as a teaching tool to explain real problems to less experienced engineers
11. The interactive nature of the approach encourages systematic, ethnographic and reflective design that could support all stages of a multidisciplinary design project

9.3.3 Recommended Extensions to the Approach

1. A practical method to evaluate the robustness of regions: apply similar tolerances in variable space, then compare the maximum and minimum values of fitness found
2. The ability to set tolerances and constraints should be made available, removing the need to manually change the objective function
3. Human input and domain knowledge should be incorporated into the clustering algorithm to ensure important information is discovered
4. Modify the clustering algorithm to include only the required variables and objectives, improving the definition of clusters and evaluation of robustness for discrete and discontinuous functions
5. Investigate the use of more sophisticated statistical and clustering techniques to inform users of sensitive variables and to discover complicated, non-linear clusters
6. The ability to evaluate the feasibility robustness of solutions should be further investigated and fully exploited
7. Investigate the use of advanced evolutionary, visualisation and clustering techniques to cope with discrete or combinatorial problems; introduce relevant artificial functions and real world problems to test the methodologies
8. Add multiobjective techniques such as Pareto optimisation to improve understanding of the trade-off surface between objectives
9. Allow users and testers to become familiar with the system by providing specific tasks with increasing complexity, this will ensure that the physical system matches the psychological understanding of the problem
10. Employ the approach in a multidisciplinary design environment; it may be necessary to integrate the ideas into other commercial systems, in which case the prospective users should participate in the design of the new system

Appendix A: The Iris Data

The Iris Data collected by Anderson (1935). Sepal and petal measurements of 150 Iris flowers (50 each of three species) are given in Table A.1 and Figure A.1.

Iris Setosa				*Iris Versicolor*				*Iris Virginica*			
S.L.	S.W.	P.L.	P.W.	S.L.	S.W.	P.L.	P.W.	S.L.	S.W.	P.L.	P.W.
5.1	3.5	1.4	0.2	7.0	3.2	4.7	1.4	6.3	3.3	6.0	2.5
4.9	3.0	1.4	0.2	6.4	3.2	4.5	1.5	5.8	2.7	5.1	1.9
4.7	3.2	1.3	0.2	6.9	3.1	4.9	1.5	7.1	3.0	5.9	2.1
4.6	3.1	1.5	0.2	5.5	2.3	4.0	1.3	6.3	2.9	5.6	1.8
5.0	3.6	1.4	0.2	6.5	2.8	4.6	1.5	6.5	3.0	5.8	2.2
5.4	3.9	1.7	0.4	5.7	2.8	4.5	1.3	7.6	3.0	6.6	2.1
4.6	3.4	1.4	0.3	6.3	3.3	4.7	1.6	4.9	2.5	4.5	1.7
5.0	3.4	1.5	0.2	4.9	2.4	3.3	1.0	7.3	2.9	6.3	1.8
4.4	2.9	1.4	0.2	6.6	2.9	4.6	1.3	6.7	2.5	5.8	1.8
4.9	3.1	1.5	0.1	5.2	2.7	3.9	1.4	7.2	3.6	6.1	2.5
5.4	3.7	1.5	0.2	5.0	2.0	3.5	1.0	6.5	3.2	5.1	2.0
4.8	3.4	1.6	0.2	5.9	3.0	4.2	1.5	6.4	2.7	5.3	1.9
4.8	3.0	1.4	0.1	6.0	2.2	4.0	1.0	6.8	3.0	5.5	2.1
4.3	3.0	1.1	0.1	6.1	2.9	4.7	1.4	5.7	2.5	5.0	2.0
5.8	4.0	1.2	0.2	5.6	2.9	3.6	1.3	5.8	2.8	5.1	2.4
5.7	4.4	1.5	0.4	6.7	3.1	4.4	1.4	6.4	3.2	5.3	2.3
5.4	3.9	1.3	0.4	5.6	3.0	4.5	1.5	6.5	3.0	5.5	1.8
5.1	3.5	1.4	0.3	5.8	2.7	4.1	1.0	7.7	3.8	6.7	2.2
5.7	3.8	1.7	0.3	6.2	2.2	4.5	1.5	7.7	2.6	6.9	2.3
5.1	3.8	1.5	0.3	5.6	2.5	3.9	1.1	6.0	2.2	5.0	1.5
5.4	3.4	1.7	0.2	5.9	3.2	4.8	1.8	6.9	3.2	5.7	2.3
5.1	3.7	1.5	0.4	6.1	2.8	4.0	1.3	5.6	2.8	4.9	2.0
4.6	3.6	1.0	0.2	6.3	2.5	4.9	1.5	7.7	2.8	6.7	2.0
5.1	3.3	1.7	0.5	6.1	2.8	4.7	1.2	6.3	2.7	4.9	1.8
4.8	3.4	1.9	0.2	6.4	2.9	4.3	1.3	6.7	3.3	5.7	2.1
5.0	3.0	1.6	0.2	6.6	3.0	4.4	1.4	7.2	3.2	6.0	1.8
5.0	3.4	1.6	0.4	6.8	2.8	4.8	1.4	6.2	2.8	4.8	1.8
5.2	3.5	1.5	0.2	6.7	3.0	5.0	1.7	6.1	3.0	4.9	1.8
5.2	3.4	1.4	0.2	6.0	2.9	4.5	1.5	6.4	2.8	5.6	2.1
4.7	3.2	1.6	0.2	5.7	2.6	3.5	1.0	7.2	3.0	5.8	1.6
4.8	3.1	1.6	0.2	5.5	2.4	3.8	1.1	7.4	2.8	6.1	1.9
5.4	3.4	1.5	0.4	5.5	2.4	3.7	1.0	7.9	3.8	6.4	2.0
5.2	4.1	1.5	0.1	5.8	2.7	3.9	1.2	6.4	2.8	5.6	2.2
5.5	4.2	1.4	0.2	6.0	2.7	5.1	1.6	6.3	2.8	5.1	1.5
4.9	3.1	1.5	0.1	5.4	3.0	4.5	1.5	6.1	2.6	5.6	1.4
5.0	3.2	1.2	0.2	6.0	3.4	4.5	1.6	7.7	3.0	6.1	2.3
5.5	3.5	1.3	0.2	6.7	3.1	4.7	1.5	6.3	3.4	5.6	2.4
4.9	3.1	1.5	0.1	6.3	2.3	4.4	1.3	6.4	3.1	5.5	1.8
4.4	3.0	1.3	0.2	5.6	3.0	4.1	1.3	6.0	3.0	4.8	1.8
5.1	3.4	1.5	0.2	5.5	2.5	4.0	1.3	6.9	3.1	5.4	2.1

...continued

5.0	3.5	1.3	0.3	5.5	2.6	4.4	1.2	6.7	3.1	5.6	2.4
4.5	2.3	1.3	0.3	6.1	3.0	4.6	1.4	6.9	3.1	5.1	2.3
4.4	3.2	1.3	0.2	5.8	2.6	4.0	1.2	5.8	2.7	5.1	1.9
5.0	3.5	1.6	0.6	5.0	2.3	3.3	1.0	6.8	3.2	5.9	2.3
5.1	3.8	1.9	0.4	5.6	2.7	4.2	1.3	6.7	3.3	5.7	2.5
4.8	3.0	1.4	0.3	5.7	3.0	4.2	1.2	6.7	3.0	5.2	2.3
5.1	3.8	1.6	0.2	5.7	2.9	4.2	1.3	6.3	2.5	5.0	1.9
4.6	3.2	1.4	0.2	6.2	2.9	4.3	1.3	6.5	3.0	5.2	2.0
5.3	3.7	1.5	0.2	5.1	2.5	3.0	1.1	6.2	3.4	5.4	2.3
5.0	3.3	1.4	0.2	5.7	2.8	4.1	1.3	5.9	3.0	5.1	1.8

Table A.1: The Iris Data, 50 examples of each species are given. (S.L.) sepal length and sepal width (S.W.), petal length (P.L.) and petal width (P.W.) all measured in centimetres (continued from previous page).

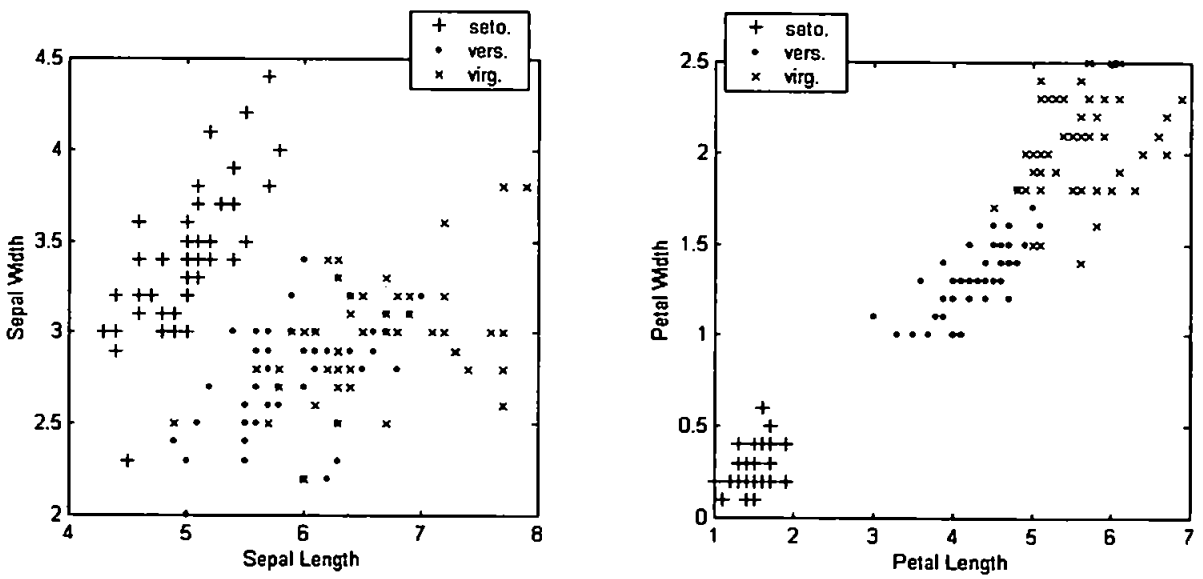


Figure A.1: Visualisation of the Iris Data: *Iris Setosa* (seto.) is linearly separable from *Iris Versicolor* (vers.) and *Iris Virginica* (virg.).

Appendix B: Test Function Definitions

Full definitions of the test functions are given in Section 6.2 and Table 6.1, apart from the transformation matrices in Test_2. Listed in the order given in Table 6.2, the matrices are:

Peak 1: 0 degrees :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Peak 2: 45 degrees :

$$\begin{bmatrix} -0.5 & 0.5 & 0.5 & 0.5 \\ 0.5 & -0.5 & 0.5 & 0.5 \\ 0.5 & 0.5 & -0.5 & 0.5 \\ 0.5 & 0.5 & 0.5 & -0.5 \end{bmatrix}$$

Peak 3: 30 degrees :

$$\begin{bmatrix} -0.\dot{6} & 0.\dot{3} & 0.\dot{6} & 0.\dot{3} \\ 0.\dot{6} & -0.\dot{3} & 0.\dot{6} & 0.\dot{3} \\ 0.\dot{6} & 0.\dot{3} & -0.\dot{6} & 0.\dot{3} \\ 0.\dot{6} & 0.\dot{3} & 0.\dot{6} & -0.\dot{3} \end{bmatrix}$$

The definition of each peak in each test function is affected by the position of the other peaks in the function, because of the interaction of peaks. Section 6.3 introduced a definition of quality for the engineering design task; the ideal regions that users and algorithms were intended to identify were shown in Figures 6.3-6.5 (for Test_3 a subset were shown). The actual height of each desired peak and the hypervolume of each corresponding were given in Table 6.2. The hypervolume is the product of the width of the desired peak in each variable (or component of a transformed coordinate system given above), assumed to be symmetrical about the position of maximum fitness in each variable (or 'centre' of each region). The location of the actual centres and the minimum and maximum values of each ideal region were calculated using numerical methods in MATLAB; they are given in Tables B.1 and B.2

	Peak No.	var 1	var 2	var 3	var 4	var 5
Test_1	1	3.0008	3.0006	3.0004	6.9993	6.9995
	2	7.0002	6.9996	3.0010	3.0004	3.0011
	3	2.0276	5.0110	2.0055	4.9889	2.0055
	4	7.9990	5.0019	7.9953	4.9980	7.9953
Test_2	1	3.0000	3.0000	3.0000	3.0000	
	2	7.0000	7.0000	7.0000	7.0000	
	3	2.0000	8.0000	2.0000	8.0000	
Test_3	1	3.0646	3.0646	3.0645	3.0645	
	2	3.0646	3.0645	3.0645	6.9565	
	3	3.0646	3.0645	6.9565	3.0645	
	4	3.0646	6.9565	3.0645	3.0645	
	5	6.9565	3.0646	3.0645	3.0645	
	6	3.0645	3.0646	6.9565	6.9565	
	7	3.0645	6.9565	3.0646	6.9565	
	8	3.0645	6.9565	6.9565	3.0646	
	9	6.9565	3.0645	6.9565	3.0646	
	10	6.9565	6.9565	3.0645	3.0646	
	11	6.9565	3.0645	3.0646	6.9565	
	12	6.9565	6.9565	6.9565	3.0645	
	13	6.9565	6.9565	3.0645	6.9565	
	14	6.9565	3.0645	6.9565	6.9565	
	15	3.0645	6.9565	6.9565	6.9565	
	16	6.9565	6.9565	6.9566	6.9566	

Table B.1: The actual centres (location of maximum fitness) of each desired region in the test functions, given in the original variable space. Compare with the centres used in the construction of the test function given in Table 6.1.

	Peak No.	var 1		var 2		var 3		var 4		var 5	
		min	max	min	max	min	max	min	max	min	max
Test_1	1	2.48	3.52	2.48	3.52	2.48	3.52	6.48	7.52	6.48	7.52
	2	5.89	8.11	5.89	8.11	1.89	4.11	1.89	4.11	1.89	4.11
	3	1.52	2.54	4.50	5.52	1.50	2.51	4.48	5.50	1.50	2.51
	4	6.91	9.09	3.91	6.09	6.90	9.09	3.91	6.09	6.90	9.09
Test_2	1	1.75	4.25	2.58	3.42	1.75	4.25	2.58	3.42		
	2*	5.75	8.25	6.58	7.42	5.75	8.25	6.58	7.42		
	3*	0.60	3.40	7.53	8.47	0.60	3.40	7.53	8.47		
Test_3	1	1.31	4.82	1.31	4.82	1.31	4.82	1.31	4.82		
	2	1.31	4.81	1.31	4.81	1.31	4.81	5.68	8.23		
	3	1.31	4.81	1.31	4.81	5.68	8.23	1.31	4.81		
	4	1.31	4.81	5.68	8.23	1.31	4.81	1.31	4.81		
	5	5.68	8.23	1.31	4.81	1.31	4.81	1.31	4.81		
	6	1.32	4.81	1.32	4.81	5.68	8.23	5.68	8.23		
	7	1.32	4.81	5.68	8.23	1.32	4.81	5.68	8.23		
	8	1.32	4.81	5.68	8.23	5.68	8.23	1.32	4.81		
	9	5.68	8.23	1.32	4.81	5.68	8.23	1.32	4.81		
	10	5.68	8.23	5.68	8.23	1.32	4.81	1.32	4.81		
	11	5.68	8.23	1.32	4.81	1.32	4.81	5.68	8.23		
	12	5.68	8.23	5.68	8.23	5.68	8.23	1.32	4.81		
	13	5.68	8.23	5.68	8.23	1.32	4.81	5.68	8.23		
	14	5.68	8.23	1.32	4.81	5.68	8.23	5.68	8.23		
	15	1.32	4.81	5.68	8.23	5.68	8.23	5.68	8.23		
	16	5.68	8.23	5.68	8.23	5.68	8.23	5.68	8.23		

*limits defined in transformed coordinate system

Table B.2: Maximum and minimum limits of the desired regions. Each region is assumed to be symmetrical in each variable around the centres given in Table B.1. For Test_2 the values are given in the transformed coordinate system after translating the original variables to the corresponding centre (see previous page).

Appendix C: Materials given to Users before Evaluation Experiments

C.1: Introduction to the System and Tests

C.2: The Engineering Design Task

C.3: Picture of the Two-Dimensional Himmelblau Function

C.4: Questionnaire for each Test Function

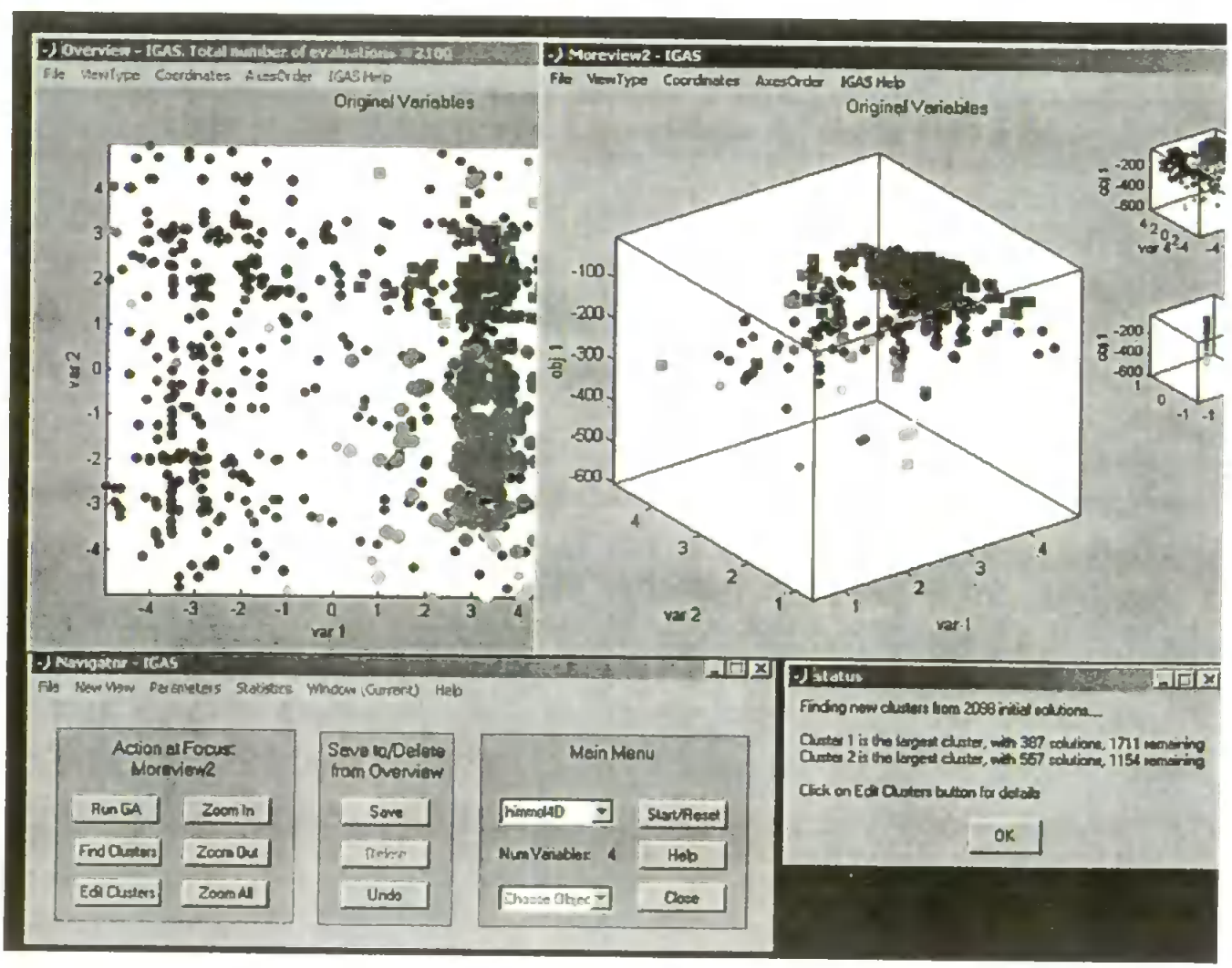
Interactive Genetic Algorithm System (IGAS)

A Graphical User Interface (GUI) designed for visualisation of multivariate data and to help solve engineering design tasks.

The system can be started by typing '*startigas*' or '*navigator*' in the Matlab Command Window at any time. All the functionality of the system can be accessed through the GUI because the Command Window is used only to display help and statistical information. Help is also available on all windows within the IGAS and is given in this html format, there is a useful **Find in page** facility at the top of the help file.

The first screen you will see is the Navigator Window. To start a session you will need to choose an objective function to optimise from the **Test Function** popupmenu. When '*Test Function*' is visible in the popupmenu, pressing the **Start/Reset** button (also on the **File** menu) will run the first objective function in the list. During a session the **Start/Reset** button will close all windows and restart using the objective shown in the **Test Function** popupmenu.

Once a session has been started the data will be shown on the **Overview Window**. Further information and data can be accessed and displayed on **Moreview Windows**. The screen shot below shows a typical display during an IGAS session.



Any problems or questions don't hesitate to ask the GUI designer (Ian Packham).

Additional Information for those taking part in the Evaluation Experiments

The first two objective functions are examples to give you time to get used to the system, the others are the test functions that will evaluate how well you and the system work together. The Engineering Design Task for the examples and test functions should have been handed to you in printed format, but is also available on screen by typing '*open engdesigntask.html*' in the Matlab Command Window.

For the purposes of this experiment the **Start/Reset** facility will become disabled after 5,000 objective function evaluations with the genetic algorithm (GA). There is also a limit on the total number of function evaluations available as indicated on the **Overview Window** - after this the **Run GA** button will be disabled, but you can continue viewing and editting the clusters until satisfied. You can save the data to file yourself at any time. At the end of the session you will be asked to fill in a short questionnaire before starting the next one. To see a sample questionnaire, type '*engdesignquest*' in the Command Window.

Note: the clustering tool does not evaluate the objective function so can be freely used between genetic algorithm runs.

Engineering Design Task

For each objective function locate regions of good solutions using the genetic algorithm (GA) and clustering tools. Try to identify those regions with maximum **objective value** and low **sensitivity**.

The test functions are all maximisation problems with a single **objective value** (sometimes called the fitness of the solution) that the GA tries to optimise. A number of solutions will be generated forming clusters that sample regions of the search space. When you have finished evaluating each objective function you will be asked to complete a questionnaire that includes ranking the clusters you have found in order of **quality** (see definitions below).

Using the tools on the interface, identify the main clusters within the seach space and try to define the variable width of each cluster such that the tolerance level is satisfied. Ignore data you consider unimportant.

Definition of Sensitivity or Robustness

A region is non-sensitive (robust) if there is no dramatic change in objective value as variables are perturbed away from the local maximum. Thus if all the solutions inside a region cover a large area (or hyper volume) and the fitness of the solutions do not fall below a certain **tolerance** value the region is considered robust.

Tolerance Guideline

For the purposes of this experiment the minimum objective value within each region should be more than 50% of its local maximum. Note: the global mimimun of all test functions is zero.

Definition of Quality

The quality of a region is a trade off between the maximum fitness and the robustness within a region. A region with high fitness and low sensitivity is of the highest quality. The true quality of a region can be measured analytically using the following definition:

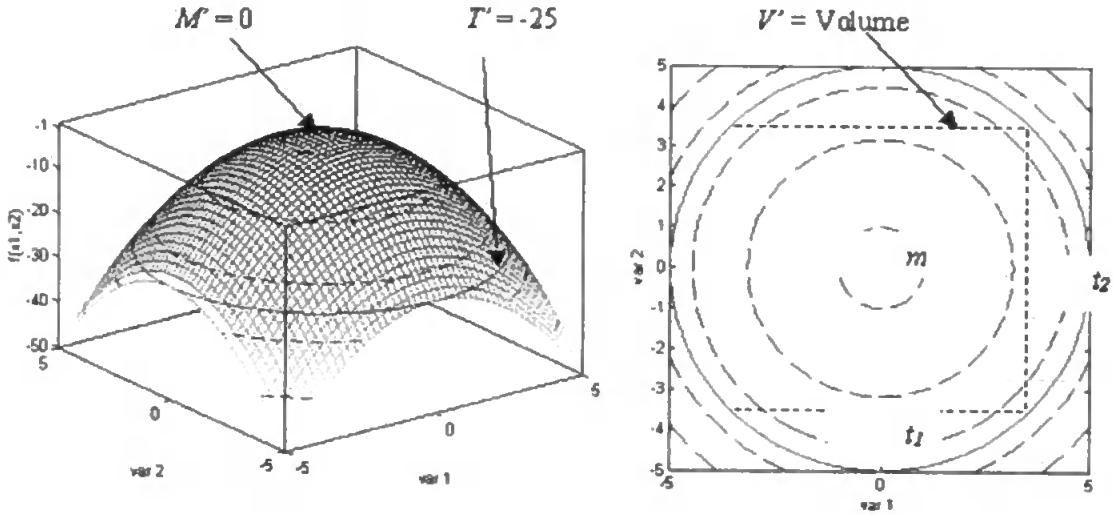
For an objective function $f(x)$, the quality of an optimum is given by:

$Q = M * V$ where Q = Quality of region,
M = Relative value of optimum,
V = Relative hypervolume of shape at tolerance level T.

For simplicity V is defined thus:

$V = \prod_{i=1}^{n} t_i$ where n = Number of variables.
If m_i = Position of optimum in i^{th} variable
and T = Tolerance level, find largest $2*n$ sided shape containing m such that $f(x) > T$,
then t_i = Length of the i^{th} side of the shape.

In the following example of **two** variables the **tolerance** required T is 50% of the optimum. The line defining the tolerated region is a circle, so the **largest** 4-sided **shape** that fits inside the circle is a square. (M' and V' are actual values of optimum and hypervolume; normalise to find M and V relative to overall limits).



The system provides statistics such as the 'Relative Volume in Variable Space' and 'Ratio of MinFit to MaxFit' to help you evaluate the quality of the clusters you have found. However the true quality of the corresponding regions may be different from the apparent quality of the clusters. For each test function the centres of the main regions are within the default limits: [0,10] for each variable. **The minimum object value for each test functions is zero.**

Reminders (from startigas)

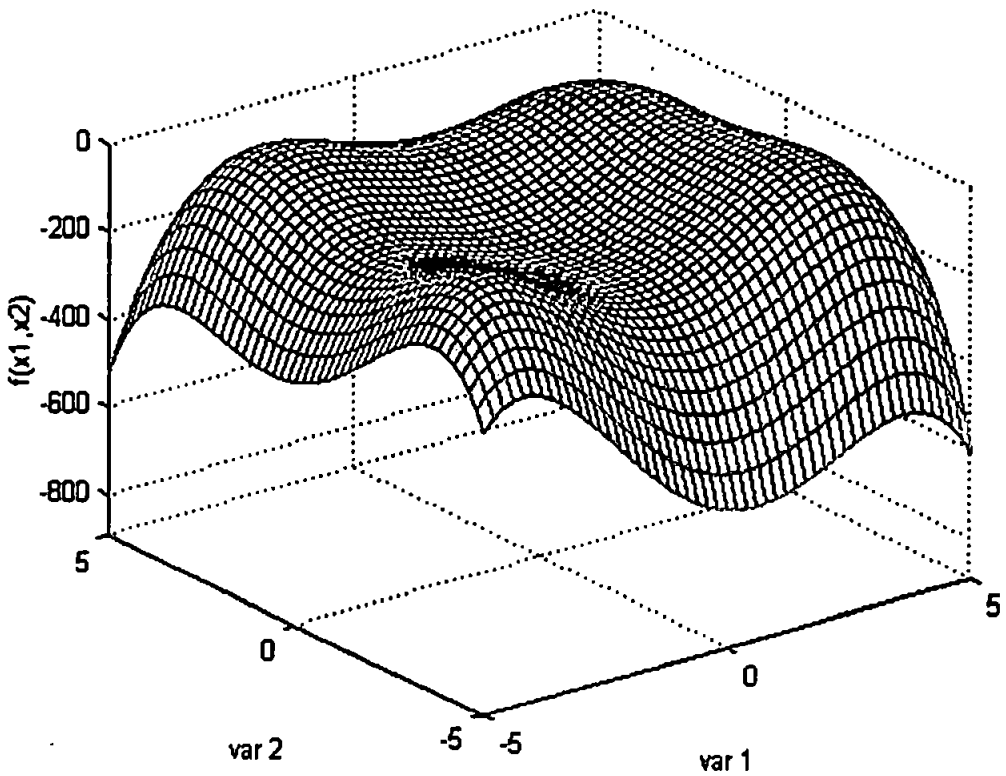
There is a limit on the total number of function evaluations available; you will be warned when the system is about to automatically end the session and save the data to file.

Note: the clustering tool does not evaluate the objective function so can be freely used between genetic algorithm runs.

To see a sample questionnaire, type 'engdesignquest' in the Command Window.

A Function: Modified Himmelblau 2D/4D/8D...

$$f(x_1, x_2) = -(x_1^{\,2} + x_2 - 11)^2 - (x_1 + x_2^{\,2} - 7)^2$$



Questionnaire for Test Function Evaluation

Your Name: _____

Test Function Name: _____

Write down the number of clusters you have identified: _____
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: _____ _____ _____ _____ _____ _____
   Quality (1 – 5): _____ _____ _____ _____ _____ _____

   Cluster Name: _____ _____ ...
   Quality (1 – 5): _____ _____ ...

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

   Cluster Name: _____ _____ _____ _____ _____ _____
   Conf. (1 – 5): _____ _____ _____ _____ _____ _____

   Cluster Name: _____ _____ ...
   Conf. (1 – 5): _____ _____ ...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): _____

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): _____

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

Appendix D: Questionnaires Returned by Users

D.1: Results and Feedback from User_1

D.2: Results and Feedback from User_2

D.3: Results and Feedback from User_3

D.4: Results and Feedback from User_4

Note: Real names have been removed from questionnaires and substituted by those used in the text.

User Name	Degree / Postgraduate Degree Subject(s)	Profession / Research Area	Engineering Design*	Evolutionary Algorithms*
User_1	Computer Engineering	Visual Neuroscience	Average	High
User_2	Psychology & Sociology / Computational Intelligence	Computational Intelligence	Average	Very High
User_3	Physiology, Pharmacology & Psychology / Audiology	Audiology / Auditory Neuroscience	Very Low	Very Low
User_4	Cognitive Science (Computer Science & Linguistics)	Evolution of Language	Low	Very High

*Self-assessed knowledge / experience rating

Table D.1: Participant educational background, current research area and self-assessed experience of engineering design and evolutionary computing

Questionnaire for Test Function Evaluation

Your Name: User_1

Test Function Name: TEST 1

Write down the number of regions you have identified: 3
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	1	2	3	____	____	____
Value (1 – 5):	2	4	5	____	____	____

Additional: _____ _____ …
Value (1 – 5): _____ _____ …

2. **Rate your confidence for each quality measure you have given**

Cluster Name:	1	2	3	____	____	____
Value (1 – 5):	5	5	5	____	____	____

Additional: _____ _____ …
Value (1 – 5): _____ _____ …

3. **How certain are you that you found all the important regions?**

Certainty (1 – 5): 1

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

Usefulness (1 – 5): 5

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

• When saving data to the Overview window it recalculated the range of variables again and I needed to find the region I was looking at before, which took me a long time.
• I have checked this comment in the second test and I think the design is quite good

Questionnaire for Test Function Evaluation

Your Name: User_1

Test Function Name: TEST 2

Write down the number of regions you have identified: 5
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	1	2	3	4	5	6
Value (1 – 5):	2	1	3	/	4	3

Additional:			...
Value (1 – 5):			...

2. **Rate your confidence for each quality measure you have given**

Cluster Name:	1	2	3	5	6	
Value (1 – 5):	5	5	2	5	5	

Additional:			...	
Value (1 – 5):			...	...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 3

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

   Usefulness (1 – 5): 5

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

- A tool to try to expand the scope of every variable defining a cluster may be useful to refine the putative boundaries

Questionnaire for Test Function Evaluation

Your Name: User_1

Test Function Name: Test 3

Write down the number of regions you have identified: 6
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	1	2	3	4	5	6
Value (1 – 5):	2	5	3	4	5	4

Additional:	____	____	...
Value (1 – 5):	____	____	...

2. **Rate your confidence for each quality measure you have given**

Cluster Name:	1	2	3	4	5	6
Value (1 – 5):	5	5	5	5	5	5

Additional:	____	____	...
Value (1 – 5):	____	____	...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 4

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

   Usefulness (1 – 5): 5

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

In the scatter plot matrix the column showing the objective value is very useful (also the row that shows this same value). Perhaps would be useful a button to "center" the regions, so that it becomes easier to zoom in

Questionnaire for Test Function Evaluation

Your Name: User_2

Test Function Name: test 1

Write down the number of regions you have identified: 2
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: 4 5 ____ ____ ____ ____
   Value (1 – 5): 4 13 ____ ____ ____ ____

   Additional: ____ ____ …
   Value (1 – 5): ____ ____ …

2. **Rate your confidence for each quality measure you have given** generations made?

   Cluster Name: 4 ____ ____ ____ ____ ____
   Value (1 – 5): ~~4~~ 5 3 ____ ____ ____ ____

   Additional: ____ ____ …
   Value (1 – 5): ____ ____ …

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 3

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

   Usefulness (1 – 5): 4

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

   1) Smaller data points for more accurate representation of the search that has been done.
   2) Proper representation of dimensions on the data point ie variable sizes for the z axis
   3) Selectable zones on more than just the 2d scatter (quite frustrating having to go back to this)

Questionnaire for Test Function Evaluation

Your Name: User_2

Test Function Name: test2

Write down the number of regions you have identified: 3
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	2	3	5				
Value (1 – 5):	2	4	2				
Additional:			...				
Value (1 – 5):			...				

2. **Rate your confidence for each quality measure you have given**

Cluster Name:	2	3	5				
Value (1 – 5):	3	2	4				
Additional:			...				
Value (1 – 5):			...				

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 5

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

   Usefulness (1 – 5): 4

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

Questionnaire for Test Function Evaluation

Your Name: User_2

Test Function Name: Test3

Write down the number of regions you have identified: 4
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	2	3	5	6	____	____
Value (1 – 5):	4	4	3	3	____	____
Additional:	____	____	...			
Value (1 – 5):	____	____	...			

2. **Rate your confidence for each quality measure you have given**

Cluster Name:	2	3	5	6	____	____
Value (1 – 5):	4	3	3	3	____	____
Additional:	____	____	...			
Value (1 – 5):	____	____	...			

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 4

4. **Rate the usefulness of the system in helping to achieve your goals(1 – 5)**

   Usefulness (1 – 5): 4

5. **Use the space below (overleaf) to enter any comments about the system. Which features were the most/least useful or suggest other tools that may have helped you complete the task.**

Questionnaire for Test Function Evaluation

Your Name: User_3

Test Function Name: TEST 1.

Write down the number of clusters you have identified: _____
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: 3 _____ _____ _____ _____ _____
   Quality (1 – 5): 4 _____ _____ _____ _____ _____

   Cluster Name: _____ _____ ...
   Quality (1 – 5): _____ _____ ...

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

   Cluster Name: 5 _____ _____ _____ _____ _____
   Conf. (1 – 5): 4 _____ _____ _____ _____ _____

   Cluster Name: _____ _____ ...
   Conf. (1 – 5): _____ _____ ...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 3

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): 4

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

1. PROBS ≡ VOL (> 500% ?)

2.

Questionnaire for Test Function Evaluation

Your Name: User_3

Test Function Name: TEST 2.

Write down the number of clusters you have identified: 1, 2, 3.
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: 2 | 3 | 1 | ____ | ____ | ____
   Quality (1 – 5): 3 | 2 | 2 | ____ | ____ | ____

   Cluster Name: ____ | ____ | ...
   Quality (1 – 5): ____ | ____ | ...

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

   Cluster Name: ~~2~~ | 3 | 1 | ____ | ____ | ____
   Conf. (1 – 5): 3 | 2 | 2 | ____ | ____ | ____

   Cluster Name: ____ | ____ | ...
   Conf. (1 – 5): ____ | ____ | ...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 2

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): 3

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

- Diff to visualise orig space VS. clusters & zoom VS. as comparisons.
- Excellent to orientate & easy to familiarise with the system.

Questionnaire for Test Function Evaluation

Your Name: User_4

Test Function Name: test 1

Write down the number of clusters you have identified: 3
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: 2 | 1 | 7 | ____ | ____ | ____
   Quality (1 – 5): 4 | 3 | 1 | ____ | ____ | ____

   Cluster Name: ____ | ____ | ...
   Quality (1 – 5): ____ | ____ | ...

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

   Cluster Name: 2 | 1 | 7 | ____ | ____ | ____
   Conf. (1 – 5): 4 | 4 | 5 | ____ | ____ | ____

   Cluster Name: ____ | ____ | ...
   Conf. (1 – 5): ____ | ____ | ...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 3

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): 4

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

Some functions/processes could be automated or grouped in some way.

Questionnaire for Test Function Evaluation

Your Name: User_4

Test Function Name: test 2

Write down the number of clusters you have identified: 2
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

   Cluster Name: 9 2 ____ ____ ____ ____
   Quality (1 – 5): 3 2 ____ ____ ____ ____

   Cluster Name: ____ ____ ...
   Quality (1 – 5): ____ ____ ...

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

   Cluster Name: 9 2 ____ ____ ____ ____
   Conf. (1 – 5): 4 4 ____ ____ ____ ____

   Cluster Name: ____ ____ ...
   Conf. (1 – 5): ____ ____ ...

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 5

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): 4

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

Questionnaire for Test Function Evaluation

Your Name: User_4

Test Function Name: test 3

Write down the number of clusters you have identified: 4
(This is every 'Cluster' defined except 'All Data 0' -
use Edit Clusters now to change definition of regions)

Answer the following questions giving a rank 1 - 5 where *1 is low* and *5 is high*:

1. **Give a quality or preference measure for each region/cluster:**

Cluster Name:	1	2	4	6		
Quality (1 – 5):	4	3	3	2		
Cluster Name:			...			
Quality (1 – 5):			...			

2. **Rate your confidence in each quality measure given (e.g. low confidence implies more time and resources are required to confirm the measure):**

Cluster Name:	1	2	4	6		
Conf. (1 – 5):	4	4	4	3		
Cluster Name:			...			
Conf. (1 – 5):			...			

3. **How certain are you that you found all the important regions?**

   Certainty (1 – 5): 4

4. **Rate the usefulness of the system in helping to achieve your goals:**

   Usefulness (1 – 5): 4

5. Use the space below to enter any comments about the system. For example which features were the most/least useful or suggest other tools that may have helped you complete the task. Did you find the experience enjoyable or frustrating?

crashed!

Appendix E: Feedback from Real World Case Studies

Questionnaire for Engineering Design System Evaluation

Your Name: MARTIN BURTHWICK

Engineering Problem Name: RAINFALL - RUNOFF MODEL

Please answer the following questions about the system and what it taught you about the engineering design problem. Please include any specific comments in the space provided.

1. **Did the system provide new or novel solutions to the problem?**

   Yes ✓ **No** _____ **Maybe** _____ **N/A** _____

   Comments: Helped 'home in' on candidate solution, some of which were subsequently used to seed a local search ('Solver' in Excel)

2. **Did the system help you evaluate the robustness of the solutions found?**

   Yes ✓ **No** _____ **Maybe** _____ **N/A** _____

   Comments: Useful to see 'isolated' good solutions - likely to be sensitive to small changes in governing parameter values (saves having to test separately, e.g. in Excel).

3. **Did the visual displays help you understand the problem?**

   Yes ✓ **No** _____ **Maybe** _____ **N/A** _____

   Comments: Particularly the improved interface (actual model parameter names & plots of predicted/observed data).

4. **Were the interactive features (Run GA/Clustering/Zoom etc.) useful?**

   Yes ✓ **No** _____ **Maybe** _____ **N/A** _____

   Comments: Useful to be able to investigate 'interesting' regions in a visually guided way - otherwise one is liable to blind searching which can be inefficient

5. **Did the system help you achieve your goals (is it relevant to engineering design)?**

   Yes ✓ **No** _____ **Maybe** _____ **N/A** _____

   Comments: Helped in both confirming prior physical knowledge of river catchment behaviour (e.g. diminishing influence of more historic data on current streamflow) & happy to rely on candidate solution for local search (as opposed to using long generation GA runs with

PTO

6. **Please use the space below to enter general comments about the system. For example which features were the most/least useful or suggest further tools are needed to help understand the engineering design problem.**

Questionnaire for Engineering Design System Evaluation

Your Name: Dave Easterbrook.

Engineering Problem Name: RC Column Optimisation Particularly Visualisation of Design Space

Please answer the following questions about the system and what it taught you about the engineering design problem. Please include any specific comments in the space provided.

1. **Did the system provide new or novel solutions to the problem?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments: Definitely. Enables user to see all solutions good & bad

2. **Did the system help you evaluate the robustness of the solutions found?**

   Yes____ **No** ✓ **Maybe**____ **N/A**____

   Comments: As a practitioner I was making observations for myself.

3. **Did the visual displays help you understand the problem?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments: Especially if the variable increase and the problem become more difficult and discrete.

4. **Were the interactive features (Run GA/Clustering/Zoom etc.) useful?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:

5. **Did the system help you achieve your goals (is it relevant to engineering design)?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:

PTO

6. **Please use the space below to enter general comments about the system. For example which features were the most/least useful or suggest further tools are needed to help understand the engineering design problem.**

Most liked was the graphical displays of solutions and the link to each solution.

Questionnaire for Engineering Design System Evaluation

Your Name: COLIN SOUTHCOMBE

Engineering Problem Name: BI AXIAL COLUMN DESIGN

Please answer the following questions about the system and what it taught you about the engineering design problem. Please include any specific comments in the space provided.

1. **Did the system provide new or novel solutions to the problem?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:
   Solutions can be clearly identified. It is "new"

2. **Did the system help you evaluate the robustness of the solutions found?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:
   A wide range of robustness solutions were identified which can be minimised or maximised

3. **Did the visual displays help you understand the problem?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:
   Solutions to the engineering problem were clear.

4. **Were the interactive features (Run GA/Clustering/Zoom etc.) useful?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:
   Very useful.

5. **Did the system help you achieve your goals (is it relevant to engineering design)?**

   Yes ✓ **No**____ **Maybe**____ **N/A**____

   Comments:

6. **Please use the space below to enter general comments about the system. For example which features were the most/least useful or suggest further tools are needed to help understand the engineering design problem.**

Initially a simple presentation would be required.

References

Abbott E.A (1884), *Flatland: A Romance of Many Dimensions*, this edition with Introduction by Alan Lightman, 1998, Penguin Books.

Agrawal R., Gehrke J., Gunopulos D. & Raghavan P. (1998), "Automatic subspace clustering of high dimensional data for data mining applications", *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD 98), SIGMOD Report*, Haas L. & Tiwary A. (eds.), Seattle, WA, June 1998, ACM Press, Vol. 27, No. 2, pp. 94-108.

Alpern B. & Carter L. (1991), "Hyperbox" in Neilson G.M. & Rosenblum L. (eds.), *Proceedings of IEEE Visualisation '91*, San Diego, California, October 1991, pp. 133-139.

Anderson E. (1935), "The Irises of the Gaspe Peninsula", *Bulletin of the American Iris Society*, Vol. 59, pp. 2-5.

Anscombe F.J. (1973), "Graphs in Statistical Analysis", *The American Statisticia*, Vol. 27, pp. 17-21.

Bäck T. (1993), "Optimal Mutation Rates in Genetic Search", *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, University of Illinois at Urbana-Champaign, 17-21 July, 1993, San Mateo, California: Morgan Kaufmann, pp. 2-8.

Bäck T. & Schwefel H.-P. (1993), "An Overview of Evolutionary Algorithms for Parameter Optimization", *Evolutionary Computation*, Vol. 1, No. 1, pp. 1-23.

Baird F., Moore C.J. & Jagodzinski A.P. (2000), "An ethnographic study of design teams at Rolls-Royce Aerospace", *Design Studies – Special Issue: Ethnographic Approaches to the Study of Engineering Design*, Jagodzinski P., Reid F. & Culverhouse P. (guest eds.), Vol. 21, No. 4, pp. 333-356.

Baker J.E. (1987), "Reducing Bias and Inefficiency in the Selection Algorithm", *Proceedings of the Second International Conference on Genetic Algorithms and their Application (ICGA-87)*, Grefenstette J.J. (ed.), Hillsdale, New Jersey: Lawrence Erlbaum Associates, pp. 14-21.

Balling R.J., Taber J.T., Day K. & Wilson S. (2000), "City Planning with a Multiobjective Genetic Algorithm and a Pareto Set Scanner", *Evolutionary Design and Manufacture, Selected Papers from ACDM'00*, Parmee I.C. (ed.), University of Plymouth, April 2000, London: Springer, pp. 237-247.

Beardah C.C (1999),. "Kernel Density Estimation Demonstrations - Help Sheet", see "MATLAB Toolbox for Density Estimation", last update: 9 September 1999: http://science.ntu.ac.uk/msor/ccb/densest.html.

Beardah C.C & Baxter M.J. (1996), "MATLAB Routines for Kernel Density Estimation and the Graphical Presentation of Archaeological Data", *Interfacing the Past, 23rd Annual Conference on Computer Applications and Quantitative Methods in Archaeology*, Leiden, March 1995, published in *Analecta Prehistorica Leidensia*, Kammermans H. & Fennema K. (eds.), Vol. 28, pp. 179-184.

Beasley D., Bull R.B. & Martin R.R. (1993), "A Sequential Niche Technique for Multimodal Function Optimization", *Evolutionary Computation*, Vol. 1, No. 2, pp. 101-125.

Becker R.A. & Cleveland W.S. (1987), "Brushing Scatterplots", *Technometrics*, Vol. 29, No. 2, pp. 127-142.

Beddow J. (1990), "Shape Coding of Multidimensional Data on a Microcomputer Display", *Proceedings of IEEE Visualization '90*, Kaufman A. (ed.), San Francisco, California, October 1990, IEEE Computer Society Press, pp. 238-246.

Beeby A.W. (1978), *The Design of Sections for Flexure and Axial Load According to CP110*, Development Report 2, Cement and Concrete Association.

Bentley P.J. (1996), "Generic Evolutionary Design of Solid Objects using a Genetic Algorithm", PhD thesis, November 1996, University of Huddersfield.

Bentley P.J. (1999), "An Introduction to Evolutionary Design by Computers", in Bentley P.J. (ed.), *Evolutionary Design by Computers*, San Francisco: Morgan Kaufmann, pp. 1-73.

Beshers C. & Feiner S. (1993), "AutoVisual: Rule-Based Design of Interactuve Multivariate Visualizations", *Computer Graphics and Applications*, Vol. 13, No. 4, July 1993, pp. 41-49. See webpage version of paper: http://www.cs.columbia.edu/graphics/projects/AutoVisual/AutoVisual.html.

Beven K.J. & Binley A.M. (1992), "The Future of Distributed Models – Model Calibration and Uncertainty Prediction", *Hydrological Processes*, Vol. 6, No. 3, pp. 279-298.

Bilchev G. & Parmee I.C. (1996), "Learning the 'Next' Dimension", *Proceedings of AISB Workshop on Evolutionary Computation, Lecture Notes in Computer Science*, Fogarty T. (ed.), University of Sussex, UK, 2 April 1996, Springer, pp. 162-174.

Biles J.A. (2001), "Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness", *Genetic and Evolutionary Computation Conference (GECCO-2001) Workshop: Non-Routine Design with Evolutionary Systems*, 7 July 2001, San Francisco, California.

Boehm B.W. (1988), "A Spiral Model of Software Development and Enhancement", *IEEE Computer*, Vol. 21, No. 5, pp. 61-72.

Bonham C.R. & Parmee I.C. (1999), "An Investigation of Exploration and Exploitation Within Cluster Oriented Genetic Algorithms COGAs)", *Proceedings of the 1999 Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann, pp. 1491-1497.

Bresler B. (1961), "Design Criteria for Reinforced Columns under Axial Load and Biaxial Bending", *Journal of the American Concrete Institute*, Vol. 32, No. 8, pp. 481-490.

Brodbeck D. & Girardin L. (2003), "Using Multiple Coordinated Views to Analyze Geo-Referenced High-Dimensional Datasets", in Roberts (2003), pp. 104-111.

BS8110 (1985), *Structural Use of Concrete: Part 1*, London British Standards Institutions.

Bucciarelli L.L. (1984), "Reflective Practice in Engineering Design", *Design Studies*, Vol. 5, No. 3, pp. 185-190.

Bucciarelli L.L. (1988), "An Ethnographic Perspective on Engineering Design", *Design Studies*, Vol. 9, No. 3, pp. 159-168.

Buja A. & Asimov D. (1986), "Grand Tour Methods: An Outline", *Computer Science and Statistics: Proceedings of the Seventeenth Symposium on the Interface*, Allen D.M. (ed.), Elsevier Science Publishers, pp. 63-67.

Buja A., Swayne D.F., Littman M.L., Dean N. & Hofmann H. (2001), "XGvis:Interactive Data Visualisation with Multidimensional Scaling", tentatively accepted for publication in the *Journal of Computational and Graphical Statistics*, available for download at: http://www.research.att.com/areas/stat/xgobi/.

Burge J. & Brown J.C. (2000), "Reasoning with Design Rationale", in Gero (2000), pp. 611-629.

Caldwell C. & Johnston V.S. (1991), "Tracking a Criminal Suspect through "Face-Space" with a Genetic Algorithm", *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, University of California, San Diego, USA, 13-16 July 1991, San Mateo, California: Morgan Kaufmann, pp. 416-421.

Card S.K., Mackinlay J.D. & Shneiderman B. (eds.) (1999a), *Readings in Information Visualization: Using Vision to Think*, San Franciso, California, Morgan Kaufmann.

Card S.K., Mackinlay J.D. & Shneiderman B. (1999b), Introduction to Ch. 3: "Interaction" in Card *et al.* (1999a), pp. 231-234.

Card S.K., Moran T.P. & Newell A. (1983), *The Psychology of Human-Computer Interaction*, Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Carpendale M.S.T., Cowperthwaite D.J. & Fracchia F.D. (1997), "Extending Distortion Viewing from 2D to 3D", *IEEE Computer Graphics and Applications*, Vol. 17, No. 4, pp. 42-51. Reprinted in Card *et al.* (1999a) pp. 368-380.

Carr D.B. (1994), "Color Perception, the Importance of Gray and Residuals, on a Choropleth Map", *Statistical Computing & Graphics*, Vol. 5 No. 1, also available online at: http://lib.stat.cmu.edu/scgn/v51/section1_6_0_1.html#SECTION06010000000000000000.

Chambers J.M., Cleveland W.S., Kleiner B. & Tukey P.A. (1983), *Graphical Methods for Data Analysis*, New York, Chapman & Hall.

Chipperfield A.J., Fleming P.J., Pohlheim H. & Fonseca C.M. (2002), "Genetic Algorithm Toolbox for Use with Matlab ®", Department of Automatic Control and Systems Engineering, University of Sheffield, last updated October 2, 2002: http://www.shef.ac.uk/~gaipp/ga-toolbox/

Cleveland W.S. (1993), *Visualising Data*, Summit, New Jersey, Hobart Press.

Collins J.J. (1999), "Visualisation of Evolutionary Algorithms using Principal Components Analysis", in GECCO (1999) pp. 99-100.

Collins T.D. (1997), "Using Software visualisation to Help Evolutionary Algorithm Users Validate their Solutions", *Proceedings of the Seventh international Conference on Genetic Algorithms (ICGA-97)*, Michigan State University, Michigan, July 1997, San Francisco, California: Morgan Kaufmann pp. 307-314.

Convertino G., Chen J., Yost B. Ryu Y.-S. & North C. (2003), "Exploring Context Switching and Cognition in Dual-View Coordinated Visualisations", in Roberts (2003), pp.55-62.

Cook D. & Buja A. (1997), "Manual Controls For High-Dimensional Data Projections", *Journal of Computational and Graphical Statistics*, Vol. 6, No. 4, pp. 464-480.

Cook D., Buja A. & Cabrera J. (1993), "Projection Pursuit Indices based on Orthogonal Function Expansions", *Journal of Computational and Graphical Statistics*, Vol. 2, No. 3, pp. 225-250.

Cook D., Buja A., Cabrera J. & Hurley C. (1995), "Grand Tour and Projection Pursuit", *Journal of Computational and Graphical Statistics*, Vol. 4, No. 3, pp. 155-172.

Coyne R.D. (1990), "Design Reasoning Without Explanations", *AI Magazine*, Vol. 11, No. 4, pp. 72-80.

Coyne R.D., Newton S. & Sudweeks F. (1993), "A Connectionist View of Creative Design Reasoning", in Gero & Maher (1993), Ch.8, pp. 177-209.

CP110 (1972), *The Structural Use of Concrete*, British Standards Instution.

Cross N. (2000), *Engineering Design Methods: Strategies for Product Design* (third edition), Chichester: John Wiley & Sons.

Darwin C. (1859), *The Origin of Species*, Edited and with an Introduction and Notes by Gillian Beer, 1996, Oxford University Press.

Davidor Y. (1991), "A Naturally Occuring Niche & Species Phenomenon: The Model and First Results", *Proceedings of the Fourth International Conference on Genetic Algorithms (ICGA-91)*, San Mateo, California: Morgan Kaufmann, pp. 257-263.

Davidson J.W., Savic D.A. & Walters G.A. (2000), "Rainfall Runoff Modeling using a New Polynomial Regression Method", *Proceedings of the 4th International Conference on Hydroinformatics*, Iowa City, Iowa, USA, 23-27 July 2000, published on CD only, p. 8.

De Jong K. (1975), "An Analysis of the Behaviour of a Class of Genetic Adaptive Systems", (Doctoral Dissertation, University of Michigan), *Dissertation Abstracts International*, 36(10), 5140B.

Deb K. (2000), "Multi-Objective Evolutionary Optimization: Past, Present, and Future", *Evolutionary Design and Manufacture, Selected Papers from ACDM'00*, Parmee I.C. (ed.), University of Plymouth, April 2000, London: Springer, pp. 226-236.

Deb K., Agrawal S., Pratap A. & Meyaruvan T. (2000), "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II", *Lecture Notes in Computer Science*, Iss. 1917; pp. 849-858.

Deb K. & Beyer H.-G. (2001), "Self-Adaptation in Real-Parameter Genetic Algorithms with Simulated Binary Crossover", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann, pp. 172-179.

Deb K. & Goldberg D.E. (1989), "An Investigation of Niche and Species Formation in Genetic Function Optimization", *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, San Mateo, California: Morgan Kaufmann, pp. 42-50.

Du X. & Chen W. (2000), "Towards a Better Understanding of Modeling Feasibility Robustness in Engineering Design", *Transactions of the ASME, Journal of Mechanical Design*, Vol. 122, Part 4, pp: 385-394.

Dym C.L. (1994), *Engineering Design: A Synthesis of Views*, Cambridge University Press.

Eckert C., Kelly I. & Stacey M. (1999), "Interactive Generative Systems for Conceptual Design: An Empirical Perspective", *Artificial Intelligence for Engineering Design, Analysis and Manufacturing - Special Issue: Generative Systems in Design*, Vol. 13, No. 4, pp. 303-320.

Eiger G., Shamir U. & Ben-Tal A. (1994), "Optimal Design of Water Distribution Networks", *Water Resources Research*, Vol. 32, No. 6, pp. 1903-1904.

Eiger G., Shamir U. & Ben-Tal A. (1996), "Reply [to Savic & Walters (1996)]", *Water Resources Research*, Vol. 32, No. 6, pp. 1903-1904.

Eshelman L. (1991), "The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination", *Foundations of Genetic Algorithms I*, Rawlins G.J.E. (ed.), San Mateo, California: Morgan Kaufmann, pp. 265-283.

Eysenck M.W. & Keane M. (2000), *Cognitive Psychology: A Student's Handbook* (fourth edition), Psychology Press.

Fasulo D., (1999), "An Analysis of Recent Work on Clustering Algorithms", Technical Report UW-CSE-01-03-02, University of Washington, USA.

Feiner S. & Beshers C. (1990), "Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds", *Proceedings of ACM Symposium on User Interface Software and Technology (UIST '90)*, Snowbird, TY, 3-5 October 1990, pp. 76-83.

Fisher R. A. (1936), "The Use of Multiple Measurements in Taxonomic Problems", *Annals of Eugenics*, Vol. 7, pp. 179-188.

Fonseca C.M. & Fleming P.J. (1993), "Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization", *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, University of Illinois at Urbana-Champaign, 17-21 July, 1993, San Mateo, California: Morgan Kaufmann, pp. 416-423.

Fonseca C.M. & Fleming P.J. (1995), "An Overview of Evolutionary Algorithms in Multiobjective Optimization", *Evolutionary Computation*, Vol. 3, No. 1, pp. 1-16.

French M.J. (1999), *Conceptual Design for Engineers* (third edition), London: Springer-Verlag.

Friedman J. H. & Tukey J. W. (1974), "A Projection Pursuit Algorithm for Exploratory Data Analsysis", *IEEE Transactions on Computers*, Vol. C-23, No. 9, pp. 881-890.

Garfinkel S. (2000), "Undo Me", a review of the book by Raskin (2000): http://www.salon.com/tech/col/2000/06/01/undo/, copyright 2003 Salon Media Group, Inc., Salon, 22 4th Street, 16th Floor, San Fransisco, CA 941003:

GECCO (1999), "Evolutionary Computation Visualisation" in Wu A.S. (ed.) *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'99) Workshop Program*, Orlando, Florida, 13 July 1999, pp. 93-107.

Gen M. & Chen R. (1997), *Genetic Algorithms and Engineering Design*, New York: John Wiley & Sons.

Gero J.S. (1990), "Design Prototypes: A Knowledge Representation Schema for Design", *AI Magazine*, Vol. 11, No. 4, pp. 26-36.

Gero J.S. (1998), "Adaptive Systems in Designing: New Analogies from Genetics and Developmental Biology", *Adaptive Computing in Design and Manufacture (ACDM '98)*, University of Plymouth, UK, 21-23 April 1998, Parmee I.C. (ed.), London: Springer-Verlag, pp. 3-12.

Gero J.S. (ed.) (2000), *Artificial Intelligence in Design (AID '00)*, Dordrecht, The Netherlands: Kluwer Academic Publishers.

Gero J.S. & Kazakov V. (2000), "Machine Learning in Design using Genetic Engineering-Based Genetic Algorithms", *Artificial Intelligence in Design (AID'00) Workshop 6 Notes: Machine Learning in Design* convened by Duffy A. at Gero (2000), 25th June 2000, Worcester Polytechnic Institute, Worcester, MA, USA.

Gero J.S. & Kumar B. (1993), "Expanding Design Spaces through New Design Variables", *Design Studies*, Vol. 14, No. 2, pp. 210-221.

Gero J.S. & Maher M.L. (eds.) (1993), *Modeling Creativity and Knowledge-Based Creative Design*, Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Gilbert D.R., Schroeder M. & van Helden J. (2000), "Interactive visualization and exploration of relationships between biological objects", *Trends in Biotechnology*, Vol. 18, No. 12, pp. 487-494.

Gnanadesikan R. (1997), *Methods for Statistical Data Analysis of Multivariate Observations*, New York, John Wiley & Sons.

GNU (2003), "GNU General Public Licence", Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA, last updated 26 May 2003: http://www.fsf.org/licenses/gpl.html

Goldberg D.E. (1989), *Genetic Algorithms in Search, Optimization & Machine Learning*, Reading, Massachusetts: Addison-Wesley.

Gordon V.S. & Whitley D. (1993), "Serial and Parallel Genetic Algorithms as Function Optimzers", *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA-93)*, University of Illinois at Urbana-Champaign, 17-21 July, 1993, San Mateo, California: Morgan Kaufmann, pp. 177-183.

Graf J. (1996), "Interactive Evolution in Engineering Design", *Proceedings of the Second International Conference on Adaptive Computing in Engineering Design and Control (ACEDC '96)*, Parmee I.C. (ed.), University of Plymouth, UK, 26-28 March 1996, pp. 297-299.

Graf J. & Banzhaf W. (1995), "An Expansion Operator for Interactive Evolution" *Proceedings of the IEEE International Conference on Evolutionary Computation*, in Perth/Australia, Piscataway: IEEE Press, pp. 798-802.

Hart E. & Ross P.M. (2000), "Enhancing the Performance of a GA through Visualisation", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Whitley D. (ed.), Las Vegas, USA, 10-12 July 2000, Morgan Kaufmann, pp. 347-354.

Hart E. and Ross P.M. (2001), "GAVEL - A New Tool for Genetic Algorithm Visualisation", *IEEE Transactions on Evolutionary Computation*, Vol. 5, No. 4, pp. 335-348.

Harvey I. & Thompson A. (1997), "Through the Labyrinth Evolution Finds a Way: A Silicon Ridge", *Proceedings of First International Conference on Evolvable Systems (ICES'96): From Biology to Hardware, Lecture Notes in Computer Science*, Higuchi T., Iwata M., & Liu W. (eds.), Springer-Verlag, Vol. 1259, pp. 406-422.

Herdy M. (1991), "Evolution Strategies with Subjective Selection", *Proceedings of the International Conference on Evolutionary Computation – The 4th International Conference on Parallel Problem Solving from Nature (PPSN IV)*, Berlin, Germany, 22-26 September, 1996, Berlin: Springer, pp. 22-31.

Hesser J. & Männer R. (1990), "Towards an Optimal Mutation Probability for Genetic Algorithms", *Proceedings of the 1st Workshop on Parallel Problem Solving from Nature (PPSN 1)*, Schwefel H.-P. & Männer R. (eds.), Dortmund, FRG, 1-3 October 1990, in *Lecture Notes in Computer Science*, Vol. 496, pp. 23-32.

Hinneburg A., Keim D.A. & Wawryniuk M. (1999), "HD-Eye: Visual Mining of High-Dimensional Data", *IEEE Computer Graphics and Applications*, Vol. 19, No. 5, pp. 22-31.

Hoffman O., Stumptner M. & Chalabi T. (2001), "A Perspective Approach to Design", *Proceedings of the KI-2001 Workshop: AI in Planning, Scheduling and Configuration and Design*, (Joint German/Austrian Conference on Artificial Intelligence), Vienna, Austria, 18 September 2001, pp. 19-31.

Holland J.H. (1975), "Adaptation in Natural and Artificial Systems", Ann Arbor: The University of Michigan Press.

Horn J. (1997), "Multicriteria Decision Making and Evolutionary Computation", *Handbook of Evolutionary Computation*, Bäck T., Fogel D.B. & Michalewicz Z. (eds.), Institute of Physics Publishing, Bristol, UK.

Hotelling H. (1947), "Multivariate Quality Control, Illustrated by the Air Testing of Sample Bombsights", *Selected Techniques of Statistical Analysis*, Eisenhart C. *et al.* (eds.), New York: McGraw-Hill, pp. 111-184.

Hudson M.G. & Parmee I.C. (1995), "The Application of Genetic Algorithms to Conceptual Design", *AI System Support for Conceptual Design (LIWED'95)*, Sharpe J.E.E. (ed.), Charlotte Mason College, Ambleside, 27-29 March 1995, Springer, pp. 17-36.

Hutchins E.L., Hollan J.D. & Norman D.A. (1986), "Direct Manipulation Interfaces", in *User Centered System Design*, Norman D.A. & Draper S.W. (eds.), Hillsdale, New Jersey: Lawrence Erlbaum Associates, Ch. 5, pp. 87-124.

Hyvärinen A. (2003), "The FastICA Package for MATLAB", Helsinki University of Technology, Laboratory of Computer and Information Science, Neural Networks Research Centre: http://www.cis.hut.fi/projects/ica/fastica/, last updated Monday 8 September 2003.

Hyvärinen A., Hoyer P.O. & Inki M. (2001), "Topographic Independent Component Analysis", *Neural Computation*, Vol. 13, Part 7, pp. 1527-1558.

Hyvärinen A. & Oja E. (2000), "Independent Component Analysis: Algorithms and Applications". *Neural Networks*, Vol. 13 (4-5), pp. 411-430.

Inselberg A. & Dimsdale B. (1994a), "Multidimensional Lines I: Representation", *SIAM Journal on Applied Mathematics*, Vol. 54, No. 2, pp. 559-577.

Inselberg A. & Dimsdale B. (1994b), "Multidimensional Lines II: Proximity and Applications", *SIAM Journal on Applied Mathematics*, Vol. 54, No. 2, pp. 578-596.

Jackson J. E. (1991), *A User's Guide op Principal Components*, New York: John Wiley & Sons.

Jagodzinski P., Reid F.J.M., Culverhouse P., Parsons R. & Phillips I. (2000), "A study of electronics engineering design teams", *Design Studies – Special Issue: Ethnographic Approaches to the Study of Engineering Design*, Jagodzinski P., Reid F. & Culverhouse P. (guest eds.), Vol. 21, No. 4, pp. 375-402.

Jain A. K., Murty M. N. & Flynn P. J. (1999), "Data Clustering: A Review", *ACM Computing Surveys*, Vol. 31, No. 3, pp. 264-323.

Jang R. (2003), "MATLAB/SIMULINK Toolboxes and Demos", CS Dept., Tsing Hua University, Taiwan: http://neural.cs.nthu.edu.tw/jang/matlab/demo/, last updated 19 September 2003.

Jarvis R.A. & Patrick E.A. (1973), "Clustering Using a Similarity Measure Based on Shared Near Neighbors", *IEEE Transactions on Computers*, Vol. 22, No. 11, pp. 1025-1034.

Jern M., Palmberg S. Ranlöf & Nilsson A. (2003), "Coordinated Views in Dynamic Interactive Documents", in Roberts (2003), pp. 95-101.

Jo J. (1998), "Interactive Evolutionary Design System: Process and Knowledge Representation", *Fourth International Round-table Conference on Computational Models of Creative Design*, Gero, J.S. & Maher, M.L. (eds.), Heron Island, Australia, December 1998, published by University of Sydney, 1999, pp. 215-224.

Jones J.C. (1992), *Design Methods* (second edition), New York: Van Nostrand Reinhold.

Josephson J.R., Chandrasekaran B., Carroll M., Iyer N., Wasacz B., Rizzoni G., Li Q., & Erb D. A. (1998), "An Architecture for Exploring Large Design Spaces", *Proceeding of the National Conference on Artificial Intelligence*, USA: Wiley, Vol. 15, pp. 143-150.

Jutten C. & Herault J. (1991), "Blind separation of sources, Part I: AN Adaptive Algorithm Based on Neuromimetic Architecture", *Signal Processing*, Vol. 24, pp. 1-10.

Kaiser P.K. (1996), "The Joy of Visual Perception: A Web Book", York University, Toronto, Canada, last updated on April 10, 2002: http://www.yorku.ca/eye/brightne.htm.

Keller P.R. & Keller M.M. (1993), *Visual Cues: Practical Data Visualization*, IEEE Computer Society Press.

Kohonen T. (1997), *Self-Organising Maps* (second edition), Berlin: Springer.

König A. (2001), "Dimensionality Reduction and Interactive Visualization of Multivariate Data - Methods, Tools, Applications", *6th On-Line World Conference on Soft-Computing WSC6*, September 2001: http://vision.fhg.de/wsc6

Leblanc J., Ward M.O. & Wittels N. (1990), "Exploring n-Dimensional Databases", *Proceedings of IEEE Visualiszation '90*, Kaufman A. (ed.), Los Alamitos, California, IEEE Computer Society Press, pp. 230-237.

Leung Y.K. & Apperley M.D. (1994), "A Review and Taxonomy of Distortion-Orientation Presentation Techniques", *ACM Transactions on Computer-Human Interaction*, Vol. 1, No. 2, pp. 126-160. Reprinted in Card *et al.* (1999a) pp. 350-367.

Levinson S.E., Rabiner L.R., Rosenberg A.E. & Wilpon J.G. (1979), "Interactive Clustering Techniques for Selecting Speaker-Independent Reference Templates for Isolated Word Recognition", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-27, No. 2, pp. 134-141.

Louis S.J. & Tang R. (1999), "Interactive Genetic Algorithms for the Travelling Salesman Problem", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann, pp. 385-392.

Lund A. (2000), "Evolving the Shape of Things to Come: A Comparison of Interactive Evolution and Direct Manipulation for Creative Tasks", extended abstract submitted to the 3rd International Conference and Exhibition on Generative Art, Politecnico di Milano University, Milan, Italy, 14-15 December 2000, see: http://www.generativeart.com/abst2000/abst53.htm.

Lund A. (2001), "Exploring TypeSpace: A Comparative Study of Interactive Evolution and Direct Manipulation", *Human-Computer Interaction – INTERACT '01*, Hirose M. (ed.), Tokyo, Japan, July 2001, published by IOS Press, IFIP, pp. 729-730.

MacQueen J. (1967), "Some Methods for Classification and Analysis of Multivariate Observations", *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, University of California Press, pp. 281-297.

Maher M.L. (1990), "Process Models for Design Synthesis", *AI Magazine*, Vol. 11, No. 4, pp. 49-58.

Mahfoud S.W. (1992), "Crowding and Preselection Revisited", *Proceedings of the 2nd Conference on Parallel Problem Solving in Nature (PPSN '92)*, Manner R. & Manderick B. (eds.), Brussels, Belgium, 28-30 September 1992, Elsevier Science Publishers, pp. 27-36.

Mahfoud S.W. (1994), "Crossover Interactions Among Niches", *Proceedings of the First IEEE Conference on Evolutionary Computation (ICEC'94)*, Orlando, Florida, USA, June 1994, Piscataway: IEEE, pp. 188-193.

Mahfoud S.W. (1995), "A Comparison of Parallel and Sequential Niching Methods", *Proceedings of the Sixth International Conference on Genetic Algorithms (ICGA'95)*, Eshelman L.J. (ed.), San Francisco, California: Morgan Kaufmann, pp. 136-143.

Maimon O. Z. & Horowitz R. (1999), "Sufficient Conditions for Inventive Solutions", *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, Vol. 29, No. 3, pp. 349-361.

Martin A.R. & Ward M.O. (1995), "High Dimensional Brushing for Interactive Exploration of Multivariate Data", *Proceedings of IEEE Visualisation '95*, Nielson G.M. & Silver D. (eds.), Los Alamitos, California: IEEE Computer Society Press, pp. 271-278.

Mathews J.D. & Rafiq M.Y (1994), "Adaptive Search for Decision Support in the Preliminary Design of Structural Systems", *Proceedings of the International Conference on Adaptive Computing in Engineering Design and Control (ACEDC '94)*, Parmee I.C. (ed.), University of Plymouth, UK, September 1994, pp. 169-175.

Mathews J.D., Rafiq M.Y. & Bullock G.N. (1996), "A Prototype for a Conceptual Structural Building Design System using the Genetic Algorithm", *Proceedings of the Second International Conference on Adaptive Computing in Engineering Design and Control (ACEDC '96)*, Parmee I.C. (ed.), University of Plymouth, UK, 26-28 March 1996, pp. 287-290.

Matthews P., Wallace K. & Blessing L. (2000), "Design Heuristics Extraction", in Gero (2000), pp. 436-453.

McGraw K. (1992), *Designing and Evaluating User Interfaces for Knowledge-Based Systems*, New York: Ellis Horwood.

Miller G.A. (1956), "The Magic Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information", *Psychological Review*, Vol. 63, pp. 81-93.

Miller B.L. & Shaw M.I. (1996), "Genetic Algorithms with Dynamic Niche Sharing for Multimodal Function Optimization", *Proceedings of the Third IEEE Conference on Evolutionary Computation*, Nagoya, Japan, May 1996, Piscataway: IEEE, pp.786-791.

Moore C.J., Miles J.C. & Rees D.W.G (1996), "Decision Support for Conceptual Bridge Design", *Artificial Intelligence in Engineering*, Vol. 11, No. 3, pp. 259-272.

Morley M.S., Atkinson R.M., Savic D.A. & Walters G.A. (2001), "GA*net*: Genetic Algorithm Platform for Pipe Network Optimisation", *Advances in Engineering Software*, Vol. 32, No. 6, pp. 467-475.

Mumford E. (1995), *Effective Systems Design and Requirements – The ETHICS Approach*, Macmillan Press Ltd.

Mumford E. (2003), *Redesigning Human Systems*, Hershey: IRM Press.

Nash I.E. & Sutcliffe I.V. (1970), "River Flow Forecasting through Conceptual Models, Part 1", *Journal of Hydrology*, Vol. 10, pp. 282-290.

Newell A. (1980), "The Knowledge Level", *AI Magazine*, Summer (1981), pp. 1-19.

Ng R.T. & Han J. (1994), "Efficient and Effective Clustering Methods for Spatial Data Mining", *Proceedings of the 20th International Conference on Very Large Data Bases*, Santiago, Chile, pp. 144-155.

Norman D.A. (1986), "Cognitive Engineering" in *User Centred System Design*, Norman D.A. & Draper S.W. (eds.), Hillsdale, New Jersey: Lawrence Erlbaum Associates, Ch. 3, pp. 31-61.

Norman D.A. (1998), *The Design of Everyday Things*, London, England: The MIT Press.

Noy P. & Schroeder M. (2001), "Introducing Signature Exploration: a Means to Aid the Comprehension and Choice of Visualization Algorithms", *12th European Conference on Machine Learning (ECML'01) & 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'01)*, Visual Data Mining Workshop, Tuesday, 4 September 2001, Freiburg, Germany.

Ochoa G., Harvey I. & Buxton H. (1999), "On Recombination and Optimal Mutation Rates", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann, pp. 488-495.

Packham I.S.J. (2000), "Pareto vs. COGA Report 01/09/00: Comparison of Adaptive Filter Values with Functions of Various Amounts of Conflict", Internal Report No. PEDC-01-00, Plymouth Engineering Design Centre, University of Plymouth, Drake's Circus, Plymouth, Devon, PL4 8AA, UK.

Packham I.S.J. and Parmee I.C. (2000), "Data Analysis and Visualisation of Cluster-Oriented Genetic Algorithm Output", *IEEE International Conference on Information Visualization (IV2000)*, Banissi E., Bannatyne M., Chen C., Khosrowshahi F., Sarfraz M. and Ursyn A. (eds.), 19-21 July 2000, London, UK, IEEE Computer Society, pp 173-178.

Packham I.S.J. & Denham S.L. (2003), "Visualisation Methods for Supporting the Exploration of High Dimensional Problem Spaces in Engineering Design", in Roberts (2003), pp. 2-13.

Padgham C.A. & Saunders J.E. (1975), *The Perception of Light and Colour*, London: G. Bell & Sons Ltd.

Pahl G. & Beitz W. (1996), *Engineering Design: A Systematic Approach*, translated and edited by Ken Wallace (second edition), Springer.

Parkinson A., Sorenson C. & Pourhassen N. (1993), "A General Approach for Robust Optimal Design", *Transactions of the ASME, Journal of Mechanical Design*, Vol. 115, No. 1, pp. 74-80.

Parmee I.C. (1996), "Cluster-Oriented Genetic Algorithms (COGAs) for the Identification of High-Performance Regions of Design Spaces", *1st International Conference on Evolutionary Computation and its Applications (EvCA96)*, Moscow, 24-27 June 1996, pp. 66-75.

Parmee I.C. & Bonham C.R. (1998), "Supporting Innovative and Creative Design using Interactive Designer / Evolutionary Computing Strategies.", *Fourth International Round-table Conference on Computational Models of Creative Design*, Gero, J.S. & Maher, M.L. (eds.), Heron Island, Australia, December 1998, published by University of Sydney, 1999, pp. 187-214.

Parmee I.C., Cvetković D., Watson A.H. & Bonham C.R. (2000), "Multiobjective Satisfaction within an Interactive Evolutionary Design Environment", *Evolutionary Computation*, Vol. 8, No. 2, pp. 197-222.

Parmee I., Cvetković D., Bonham C. & Packham I. (2001), "Introducing Prototype Interactive Evolutionary Systems for Ill-Defined, Multi-Objective Design Environments", *Advances in Engineering Software*, Elsevier Science Ltd, Vol. 32, No. 6, pp. 429-441.

Phadke M.S. (1989), *Quality Engineering using Robust Design*, Prentice-Hall International.

Pham D.T. & Yang Y. (1993a), "A Genetic Algorithm Based Preliminary Design System", *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, Vol. 207, No. 2, pp. 127-133.

Pham D.T. & Yang Y. (1993b), "Optimisation of Multi-modal Discrete Functions using Genetic Algorithms", *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, Vol. 207, No. 1, pp. 53-59.

Pohlheim H. (1996), "Selection" help file in "Users Guide" to "GEATbx: Genetic and Evolutionary Algorithm Toolbox for use with MATLAB", Version 1.91 (July 1997). http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/algselct.html.

Pohlheim H. (1999), "Visualization of Evolutionary Algorithms – Set of Standard Techniques and Multidimensional Visualization", *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, Florida, USA, 13-17 July 1999, San Francisco: Morgan Kaufmann, pp. 533-540.

Preece J., Rogers Y., Sharp H., Benyon D., Holland S. & Carey T. (1994), *Human-Computer Interaction*, Harlow, England: Addison-Wesley.

Pugh S. (1990), *Total Design, Integrated Methods for Successful Product Engineering*, Wokingham, England: Addison-Wesley.

Rafiq M. Y. (2000), "Importance of Pareto Optimum Solutions in Making Informed Decisions in Engineering Design", *Proceedings of the 8th International Computing in Civil and Building Engineering, ICCCB-VIII*, Stanford University, USA, August 2000, pp. 1325 - 1333.

Rafiq M.Y. & Southcombe C. (1998), "Genetic Algorithms in Optimal Design and Detailing of Reinforced Concrete Biaxial Columns Supported by a Declarative Approach for Capacity Checking", *Computers and Structures*, Vol. 69, pp. 443-457.

Raskin J. (2000), *The humane Interface: New directions for designing interactive systems*, Reading, Massachusetts: Addison-Wesley

Reffat R.M. & Gero J.S. (2000), "Computational Situated Learning in Design", in Gero (2000), pp. 589-610.

Roberts J. (ed.) (2003), *Proceedings of International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV2003)*, London, UK, 15 July 2003, IEEE Computer Society.

Ross P. & Corne D. (1994), "Applications of Genetic Algorithms", invited paper in *AISBQ, the Quarterly Journal of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour, Special Issue on Evolutionary Computation*, Fogarty T. (ed.), Winter 1994, Vol. 89, pp. 23-30.

Roy R. Parmee I.C. & Purchase (1996), "Integrating the Genetic Algorithm with the Preliminary Design of Gas Turbine Blade Cooling Systems", *Proceedings of the Second International Conference on Adaptive Computing in Engineering Design and Control (ACEDC '96)*, Parmee I.C. (ed.), University of Plymouth, UK, 26-28 March 1996, pp. 228-235.

Sammon J.W. (1969), "A Nonlinear Mapping for Data Structure Analysis", *IEEE Transactions on Computers*, Vol. C-18, No. 5, pp. 401-409.

Sato T. & Hagiwara M. (2001), "IDSET: Interactive Design System using Evolutionary Techniques", *Computer Aided Design*, Vol. 33, pp. 367-377.

Savic D.A. & Walters G.A. (1994), "Evolution Programs in Optimal Design of Hydraulic Networks", *Proceedings of the International Conference on Adaptive Computing in Engineering Design and Control (ACEDC '94)*, Parmee I.C. (ed.), University of Plymouth, UK, September 1994, pp. 146-150.

Savic D.A. & Walters G.A. (1996), "Comment on 'Optimal Design of Water Distribution Networks' by Gideon Eiger, Uri Shamir, and Aharon Ben-Tal", *Water Resources Research*, Vol. 32, No. 6, pp. 1899-1901.

Savic D.A., Walters G.A. & Davidson J.W. (1999), "A Genetic Programming Approach to Rainfall-Runoff Modelling", *Water Resources Management*. Vol. 13, pp. 219-231.

de Schaetzen W., Rosseau M., Savic D. & Walters G. (2000), "Analysis of the Sensitivity of the Calibration Optimisation Problem", *Water Network Modelling for Optimal Design and Management, CWS 2000,* Walters G.A. & Savic D.A. (eds.), September 2000, Exeter, UK, pp. 43-54.

Schön D.A. (1984), "Problems, Frames and Perspectives on Designing", *Design Studies*, Vol. 5, No. 3, pp. 132-136.

Schön D.A. (1988), "Designing: Rules, Types and Worlds", *Design Studies*, Vol. 9, No. 3, pp. 181-190.

Schön D.A. & Wiggins G. (1992), "Kinds of Seeing and their Functions in Designing", *Design Studies*, Vol. 13, No. 2, pp. 135-156.

Scott D.W. (1992), *Multivariate Density Estimation*, New York, John Wiley & Sons.

Shine W.B. & Eick C.F. (1997), "Visualizing the Evolution of Genetic Algorithm Search Processes", *Proceedings of 1997 IEEE International Conference on Evolutionary Computation (ICEC '97)*, University Place Hotel, Indinapolis, USA, 13-16 April 1997, pp. 367-372.

Shipman R., Shackleton M. & Harvey I. (2000), "The use of Neutral Genotype-Phenotype Mappings for Improved Evolutionary Search", *BT Technology Journal*, Vol. 18, No. 4, pp. 103-111.

Shneiderman B. (1998), *Designing the User Interface*, Reading, Massachusetts: Addison-Wesley.

Sierra A. & Corbacho F. (2000), "Reclassification as Supervised Clustering", *Neural Computation*, Vol. 12, pp. 2537-2546.

Siirtola H. (1999) "Interaction with the Reorderable Matrix ", *Proceedings of IEEE International Conference on Information Visualisation (IV'99)*, Banissi E. (ed.), London, UK, July 1999, pp. 272-279.

Siirtola H. (2000), "Direct Manipulation of Parallel Coordinates", *Proceedings of IEEE International Conference on Information Visualisation (IV2000)*, Banissi E., Bannatyne M., Chen C., Khosrowshahi F., Sarfraz M. and Ursyn A. (eds.), London, UK, 19-21 July 2000, Los Alamitos, California: IEEE Computer Society, pp. 373-378.

Siirtola H. (2003), "Combining Parallel Coordinates and the Reorderable Matrix", in Roberts (2003), pp. 63-74.

Silverman B.W. (1986), *Density Estimation for Statistics and Data Analysis*, London: Chapman and Hall.

Sim S.K. & Duffy A.H.B. (2000), "Evaluating a Model of Learning in Design using Protocol Analysis", in Gero (2000), pp. 455-477.

Smith T., Bullock S. & Bird J. (2002), "Beyond Fitness: Visualising Evolution – Workshop Overview", *ALife VIII Workshop Proceedings*, Bilotta E., Groß D., Smith T., Lenaerts T., Bullock S., Lund H.H., Bird J., Watson R., Pantano P., Pagliarini L., Abbass H., Standish R. & Bedau M. (eds.), 9-13 December 2002, University of New South Wales, Sydney, NSW, Australia, p. 99.

Smithers T. (2000), "Designing a Font to Test a Theory", in Gero (2000), pp. 3-22.

Sorooshian S. & Gupta V.K. (1995), "Model Calibration", in *Computer Models of Watershed Hydrology*, Singh V.P. (ed.), Water Resource Publications, pp. 23-68.

Spears W.M. (1999), "An Overview of Multidimensional Visualisation Techniques", in GECCO (1999), pp. 104-105.

Spence R. (2001), *Information Visualization*, Harlow, England: Addison-Wesley.

Spence R. (2003), "The Acquisition of Insight", Intelligent and Interactive Systems Group, Electrical and Electronic Engineering Department, Imperial College, Exhibition Road, London, SW7 2BT: http://www.ee.ic.ac.uk/research/information/www/Bobs.html, last accessed November 2003.

Spence R. & Tweedie L. (1998), "The Attribute Explorer: Information Synthesis Via Exploration", *Interacting with Computers*, Vol. 11, pp. 137-146.

Swayne D. F., Cook D. & Buja A. (1998), "XGobi: Interactive Dynamic Data Visualization in the X Window System", *Journal of Computational and Graphical Statistics*, Vol. 7, No. 1, pp. 113-130.

Swayne D. F., Cook D., Buja A & Lang D.T. (2001), "ggobi Manual", September 2001, see http://www.ggobi.org/Documentation.html, last updated 5 November 2001.

Taguchi G. (1986), *Introduction to Quality Engineering: Designing Quality into Products and Processes*, Asian Productivity Organisation.

Takagi H. (1996), "Interactive GA for System Optimization: Problems and Solution", *Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing (EUFIT '96)*, Aachen, Germany, 2-5 September 1996, pp. 1440-1444.

Takagi H. (2001), "Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation", *Proceedings of the IEEE*, Vol. 89, No. 9, pp. 1275-1296.

Tsutsui S. & Fujimoto Y. (1993), "Forking Genetic Algorithm with Blocking and Shrinking Modes (fGA)", *Proceedings of the Fifth International Conference on Genetic Algorithms*, Forrest S. (ed.), San Mateo, California: Morgan Kaufmann, pp. 206-213.

Tufte E.R. (1983), T*he Visual Display of Quantitative Information*, Cheshire, Connecticut, Graphics Press.

Tufte E.R. (1990), *Envisioning Information*, Cheshire, Connecticut, Graphics Press.

Tukey J.W. (1977), *Exploratory Data Analysis*, Reading, Massachusetts, Addison-Wesley.

Tweedie L., Spence R., Dawkes H. & Su H. (1996a) "The Influence Explorer - a Tool for Design", Video Proceedings of Conference on Human Factors in Computing Systems (CHI'96), 13-18 April 1996, Vancouver, British Columbia, Canada, see: http://www1.acm.org/sigs/sigchi/chi96/proceedings/videos.htm.

Tweedie L., Spence R., Dawkes H. & Su H. (1996b), "Externalizing Abstract Mathematical Models", *Proceedings of CHI'96*, ACM, pp. 406-412. Reprinted in Card *et al.* (1999a) pp. 276-284.

Tsykin E.N. (1985), "Multiple Nonlinear Statistical Models for Runoff Simulation and Prediction", *Journal of Hydrology*, Vol. 77, pp. 209-226.

Vesanto J., Himberg J., Alhoniemi E. and Parhankangas J. (2000), "SOM Toolbox for Matlab 5", Report No. A57, April 2000, Neural Networks Research Centre, Helsinki University of Technology, Finland, see: http://www.cis.hut.fi/projects/somtoolbox/, last updated 31 January 2003.

Walters G.A., Savic D.A., Thurley R.W.F, Halhal D., Kapelan Z. & Atkinson R. (1999) "Optimal Design of Water Systems using Genetic Algorithms: Some Recent Developments", *Computing and Control for the Water Industry*, Powell R. & Hindi K.S. (eds.), Baldock, Herts., UK: Research Studies Press, Iss. 2, pp. 337-344.

Wang Q.J. (1991), "The Genetic Algorithm and its Application to Calibrating Conceptual Rainfall-Runoff Models", *Water Resources Research*, Vol. 27, No. 9, pp. 2467-2471.

Ward M.O. (1994), "XmdvTool: Integrating Multiple Methods for Visualizing Multivariate Data", *Proceedings of IEEE Visualization '94*, Bergeron R.D. & Kaufman A.E. (eds.) Los Alamitos, California: IEEE Computer Society Press, pp. 326-336.

van Wijk J.J & van Liere R. (1993), "HyperSlice: Visualization of Scalar Functions of Many Variables.", *Proceedings of IEEE Visualization '93*, Nielson G.M. & Bergeron r.D. (eds.), Los Alamitos, California: IEEE Computer Society Press, pp. 119-125.

Wong P.C. & Bergeron R.D. (1997), "30 Years of Multidimensional Multivariate Visualization", in *Scientific Visualization: Overviews, Methodologies, Techniques*, Nielson G.M., Hagemn H., Müller H. (eds.), Los Alamitos, California, IEEE Computer Society Press, Ch. 1, pp. 3-33.

Wyszecki G. & Stiles W.S. (1967), *Color Science: Concepts and Methods, Quantitative Data and Formulas*, New York: John Wiley & Sons.